Programmer's Guide

NASA Technical Memorandum 4261
Part 1

GEMPAK5
Part 1—GEMPAK5 Programmer's Guide

Version 5.0

Mary L. desJardins
Goddard Space Flight Center
Greenbelt, Maryland

Keith F. Brill
NOAA/NWS/NMC
Washington, D.C.

Steven S. Schotz
General Sciences Corporation
Lanham, Maryland

NASA
National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Division
1991

CHAPTER 1

OVERVIEW

1.1 INTRODUCTION

This document is the application programmer's manual for GEMPAK. It is intended to help programmers write stand-alone programs which call GEMPAK subroutines as well as application programs which run as part of GEMPAK. The GEMPAK User's Manual provides information on running GEMPAK; the GEMPAK Installation Guide provides information on bringing up GEMPAK at a site. GEMPLT graphics subroutines are documented in the GEMPLT Programmer's Guide.

All of the GEMPAK subroutines have been written in standard FORTRAN/77 (plus DO WHILE and END DO constructs) with modularity, documentation, and extensibility as important design considerations. The code was developed on a VAX 11/780 running the VMS operating system and has been ported to several Unix machines.

GEMPAK is designed to work with the TAE (Transportable Applications Executive) user interface. For sites that do not have the TAE installed, or users who want a simpler interface, a non-TAE (NT) interface is available. The IP library is the programmer interface to both the TAE and the non-TAE versions.

1.2 ACCESS TO GEMPAK

The programmer must have the logical names GEMUSR and GEMLIB assigned to the roots of the GEMPAK user files and the GEMPAK software files respectively. GEMUSR and GEMLIB may point to the same directory. Once the assignments are made, execute the command:

@GEMLIB:PASSIGN

The following (partial) list of logical names will be assigned:

GEMOLB	-	GEMPAK object libraries
GEMTABL	-	GEMPAK tables
GEMERR	-	GEMPAK error files
GEMEXE	-	GEMPAK programs

Also, the TAE logical names required by GEMPAK will be assigned.

1.3 SUBROUTINE LIBRARIES

GEMPAK programs are built in a modular fashion using an extensive set of subroutines grouped by function into GEMPAK libraries. Each library subroutine name begins with two letters indicating the library function followed by an underscore ('_'). For example, SF_OPNF is a subroutine from the SF (surface) library which opens a surface file. Since the names of some GEMPAK subroutines may not be known to the programmer, programmer-defined subroutine names should not follow the xx_... pattern to avoid unintentional duplication of subroutine names.

Documentation for all of the program-callable GEMPAK library subroutines is included in the rest of this manual along with brief descriptions of the libraries' functions.

1.4 OBJECT CODE

The object code for the GEMPAK subroutines is contained in object libraries in a directory whose logical name is GEMOLB: . The GEMPAK library is:

GEMOLB:GEMLIB

The GEMPLT object code is located in:

GPOLB:APPL

The TAE object libraries are:

TAE$OLB:TAELIB
TAE$OLB:COTS

The GEMPLT and TAE libraries must be included only if GEMPLT or TAE

calls are made within the program being linked.

If more than one of these libraries is used, they must be included in the link command in the order given above. For example, the program GDCNTR which uses the general GEMPAK library, the GEMPLT library, and the TAE can be linked using the command:

```
LINK/EXE=GDCNTR  GDCNTR/LIB,-
                 GEMOLB:GEMLIB/LIB,-
                 GEMOLB:APPL/LIB,-
                 GEMOLB:GEMLIB/LIB,-
                 TAE$OLB:TAELIB/LIB,-
                 TAE$OLB:COTS/LIB
```

1.5 TAE

The TAE is used by GEMPAK to provide the interface between the user and the application programs. In order to receive variables from the TAE, the programmer should use the GEMPAK IP library. No TAE subroutines should be called directly; TAE subroutines cannot be mixed with the IP modules. Using the IP library allows the program to enter a 'dynamic tutor,' in which the user can enter new values for the program variables without exiting the program. Any TAE error encountered will be printed immediately by the IP subroutines.

GEMPAK programs use TAE global parameters to save default values of the input variables. These defaults are retained between programs and from one session to the next. Whenever the value of a global parameter is changed, the new value becomes the default (provided the program in which it was changed ran successfully). As an example, if the user sends output to a file in SFLIST, then enters SNLIST, the initial value of OUTPUT will be F. Global parameters are designated by a **$** as the first character.

All GEMPAK parameters have a comparable TAE global value. As a result, all parameter changes made by the user in executing one program will carry over to following programs. The global parameter default values are updated with a call to the IP_USTR subroutine.

In order for a program to receive variables from the TAE, a text file called a PDF must be written. A complete description of the PDF files is included in the TAE Programmer's Guide. The following description applies to writing PDFs for GEMPAK programs.

There are four parts needed for each PDF:

1. variable description
2. help for the program
3. level 1 help for the variables
4. level 2 help for the variables

arranged as follows:

```
PROCESS HELP=*
   [ variable description ]
END-PROC
.TITLE
   [ program title ]
.HELP
   [ program help ]
.LEVEL1
   [ level 1 help ]
.LEVEL2
   [ level 2 help ]
.END
```

The variable description must use REFGBL statements to name the global variables that are referenced in the program. $TUTOR, $MAPFIL and $RESPOND must appear in all GEMPAK programs. Then local variables are named, all of which correspond to global parameters. The command procedure PDFBLD will build the PDF file from a .PRM file which lists all the program parameters. The program help files (.HLP) and parameter help files are stored in GEMHLP.

1.6 METEOROLOGICAL PARAMETERS

Several facilities are available to the GEMPAK programmer for computing a variety of parameters. All meteorological parameters are given names in GEMPAK. All the observed quantities which may be found in surface or sounding data sets are given 4-character names. A list of the abbreviations and a description of the parameters are given as an appendix to the GEMPAK User's Guide.

The PR library is a collection of functions which can be used to compute meteorological parameters from other parameters. The PR library contains general purpose routines which can be used without reference to the other GEMPAK libraries.

The PC library is available to compute parameters automatically from the parameters given in surface and sounding data sets. In addition, the PC library will convert upper-air data to different vertical coordinate systems, and will interpolate and extrapolate data.

The DG library allows computation of diagnostic functions from gridded data. The functions are expressed as nested strings of operators and operands, allowing flexibility in calculating new quantities. In addition, in-line flags for time, level, and vertical coordinate permit the user additional freedom in defining functions. The grid diagnostics are documented in an appendix to the User's Guide.

1.7 ERROR PROCESSING

Error messages in GEMPAK programs should be printed using the subroutine ER_WMSG. The text of error messages is saved in text files called xx.ERR, where xx is the subroutine library or program name. These files must reside in the directory pointed to by GEMERR. The error file format is described in the ER library documentation.

Most library subroutines will not print error messages themselves except for the TAE errors, which are printed by the IP library, and FORTRAN file errors, which are printed from subroutines which open, read, or write to files.

1.8 I/O

All of the I/O done by GEMPAK is FORTRAN I/O. Programmers should use the data-set libraries to access data files rather than read them directly.

Programmers who find it necessary to do I/O not provided by GEMPAK subroutines can use FL_PERR to print error messages. This subroutine must be called when the error is encountered. The value for IOSTAT must be passed to it. The error will be interpreted and printed using ER_WMSG.

GEMPAK tables may be accessed using the TB library subroutines. These should be used so that the format of the table files can be changed in the future without adversely affecting existing software. The GEMPAK tables are stored in GEMTABL. If new tables are needed, the appropriate TB subroutine should be written.

1.9 GRAPHICS

All the graphics and transformation functions are provided by GEMPLT. The GG library is provided to standardize and simplify some calls to the GEMPLT subroutines. Attributes may be set using IN subroutines. Most graphics plotting calls will be made directly to the GEMPLT subroutines.

CHAPTER 2

DIAGNOSTIC GRID (DG) LIBRARY

DG_AREA	Set area for diagnostics
DG_FLNO	Retrieve file number from GFUNC
DG_GRID	Compute scalar grid
DG_INIT	Initialize diagnostic package
DG_OANG	Set orientation angle
DG_OFIL	Open multiple grid files
DG_VECR	Compute grid relative vector
DG_VECT	Compute vector grid

Diagnostic Grid (DG) Library Summary

The DIAGNOSTIC GRID package provides subroutines to perform diagnostic computations on gridded fields. DG_GRID computes scalar quantities; DG_VECR and DG_VECT compute vector quantities in grid relative and north relative coordinates, respectively.

The diagnostics package must be initialized each time new grid files are to be accessed. Usually, DG_OFIL will be called to open the grid files and perform the initialization. If only one file is to be opened, DG_INIT may be called instead.

DG_GRID, DG_VECR and DG_VECT require the date/time, vertical level, vertical coordinate and grid diagnostic function that were input by the user. Although input and intermediate grids may be either scalars or vectors, the output for DG_GRID must be a scalar, and for DG_VECR or DG_VECT must be a vector.

ERROR MESSAGES:

[DG -1] Grid size is too large.
[DG -2] Grid size is invalid.
[DG -3] GFUNC is blank.
[DG -4] Output grid is not a scalar.
[DG -5] Output grid is not a vector.
[DG -6] An operator has an incorrect number of operands.
[DG -7] Input grid ... cannot be found.
[DG -8] Input grid ... is the wrong size.
[DG -9] Operator ... has a calling sequence error.
[DG -10] Internal grid list is full; simplify function.
[DG -11] Operand ... must be a vector.
[DG -12] Operand ... must be a scalar.
[DG -13] Operand ... must be read from grid file.
[DG -14] DG_INIT has not been called.
[DG -15] Center of polar grid is not valid.
[DG -16] Map projection ... is invalid.
[DG -17] LEVEL ... must be a layer.
[DG -18] TIME ... must be a time range.
[DG -19] Operator ... is not recognized.
[DG -20] Stack is full; simplify function.
[DG -21] Stack is empty; check operands, nesting.
[DG -22] TIME ... is invalid.
[DG -23] LEVEL ... is invalid.
[DG -24] IVCORD ... is invalid.
[DG -26] Layer of layers is invalid.
[DG -27] Layer of time range is invalid.
[DG -28] No orientation vector for TANG or NORM.

[DG -29] No grid file name specified.
[DG -30] Error opening grid file.
[DG -31] Navigation is not the same as in first grid file.
[DG -32] Invalid file number.

DG Library Calls

DG_AREA (igxmin, igxmax, igymin, igymax, / iret)

DG_FLNO (gfunc, / igdfln, iret)

DG_GRID (gdattm, glevel, gvcord, gfunc, / pfunc, grid, igx, igy, time, level, ivcord, parm, iret)

DG_INIT (igdfln, rnav, lasttm, / iret)

DG_OANG (orient, / iret)

DG_OFIL (gdfile, gdoutf, / igdfln, ioutfl, iret)

DG_VECR (gdattm, glevel, gvcord, gvect, / pfunc, ugrid, vgrid, igx, igy, time, level, ivcord, parmu, parmv, iret)

DG_VECT (gdattm, glevel, gvcord, gvect, / pfunc, ugrid, vgrid, igx, igy, time, level, ivcord, parmu, parmv, iret)

2.1 DG_AREA - SET AREA FOR DIAGNOSTICS

This subroutine defines the grid subarea needed for diagnostics in an application program. If this subroutine is called, diagnostics will be computed only over this subarea so that the computations will execute faster.

DG_AREA (IGXMIN, IGXMAX, IGYMIN, IGYMAX, IRET)

Input parameters:

IGXMIN	INTEGER	Minimum x grid coordinate
IGXMAX	INTEGER	Maximum x grid coordinate
IGYMIN	INTEGER	Minimum y grid coordinate
IGYMAX	INTEGER	Maximum y grid coordinate

Output parameters:

IRET	INTEGER	Return code 0 = normal return

2.2 DG_FLNO - RETRIEVE FILE NUMBER FROM GFUNC

This subroutine returns the grid file number corresponding to the first grid file referenced in GFUNC. This number can be used to call GD_ subroutines to find the levels in a grid file.

DG_FLNO (GFUNC, IGDFLN, IRET)

Input parameters:

GFUNC	CHAR*	Input for GFUNC

Output parameters:

IGDFLN	INTEGER	Grid file number
IRET	INTEGER	Return code 0 = normal return -32 = invalid file number

2.3 DG_GRID - COMPUTE SCALAR GRID

This subroutine computes a grid diagnostic scalar quantity. The inputs for GDATTM, GLEVEL, GVCORD and GFUNC should be the values input by the user.

```
DG_GRID  ( GDATTM, GLEVEL, GVCORD, GFUNC, PFUNC, GRID,
           IGX, IGY, TIME, LEVEL, IVCORD, PARM, IRET )
```

Input parameters:

GDATTM	CHAR*	Input date/time
GLEVEL	CHAR*	Input level
GVCORD	CHAR*	Input vertical coordinate
GFUNC	CHAR*	Diagnostic function

Output parameters:

PFUNC	CHAR*	Diagnostic error string
GRID (IGX,IGY)	REAL	Output scalar grid
IGX	INTEGER	Number of points in x dir
IGY	INTEGER	Number of points in y dir
TIME (2)	CHAR*	Output date/time
LEVEL (2)	INTEGER	Output level
IVCORD	INTEGER	Output vertical coordinate
PARM	CHAR*	Output parameter name
IRET	INTEGER	Return code 3 = user typed EXIT 0 = normal return -3 = GFUNC is blank -4 = output grid not a scalar -6 = wrong number of operands -7 = grid cannot be found -8 = grid is the wrong size -9 = incorrect operands -10 = internal grid list is full -11 = operand must be a vector -12 = operand must be a scalar -13 = operand must be from grid -14 = DG_INIT not initialized -15 = polar center invalid -16 = map proj is invalid -17 = LEVEL must be a layer -18 = TIME must be a range -19 = invalid operator -20 = stack is full -21 = stack is empty -22 = TIME is invalid -23 = LEVEL is invalid -24 = IVCORD is invalid -26 = layer of layers invalid -27 = time range layer invalid

2.4 DG_INIT - INITIALIZE DIAGNOSTIC PACKAGE

This subroutine initializes the grid diagnostics package for a grid file. Note that this subroutine is called by GR_FILE. It should be called only in programs which will use DG_GRID, DG_VECT or DG_VECR, but will not open the grid file using GR_FILE. When DG_INIT is called, GR_OPEN and GR_SNAV must be called first to open the file and define the grid navigation, respectively.

In general, DG_OFIL should now be used to open files. This subroutine is included for use in older programs.

DG_INIT (IGDFLN, RNAV, LASTTM, IRET)

Input parameters:

IGDFLN	INTEGER	Grid file number
RNAV (256)	REAL	Grid navigation block
LASTTM	CHAR*	Last time in grid file

Output parameters:

IRET	INTEGER	Return code 0 = normal return -1 = grid size is too large -2 = grid size is invalid

2.5 DG_OANG - SET ORIENTATION ANGLE

This subroutine sets the orientation angle for the grid diagnostics package. This angle is usually used to determine normal and tangential components of vectors with respect to a cross section. The tangential components are along the orientation angle.

DG_OANG (ORIENT, IRET)

Input parameters:

ORIENT	REAL	Orientation angle in radians

Output parameters:

IRET	INTEGER	Return code 0 = normal return

2.6 DG_OFIL - OPEN MULTIPLE GRID FILES

This subroutine opens grid files and initializes the grid diagnostics package. It should be called whenever more than one grid file might be input. The input grid file names must be separated with a +. Only one output file name is allowed. The sum of distinct input and output files cannot exceed four.

DG_OFIL (GDFILE, GDOUTF, IGDFLN, IOUTFL, IRET)

Input parameters:

GDFILE	CHAR*	Grid file names
GDOUTF	CHAR*	Output grid file name

Output parameters:

IGDFLN	INTEGER	Grid file number for file 1
IOUTFL	INTEGER	Output grid file number
IRET	INTEGER	Return code 0 = normal return -29 = file open failure -30 = error opening file -31 = navigation not the same -33 = too many files to open -34 = more than one output file

2.7 DG_VECR - COMPUTE GRID RELATIVE VECTOR

This subroutine computes a grid diagnostic vector quantity. The u and v components returned in UGRID and VGRID are in grid relative coordinates. GDATTM, GLEVEL, GVCORD and GVECT should have the values entered by the user.

```
DG_VECR ( GDATTM, GLEVEL, GVCORD, GVECT, PFUNC, UGRID,
          VGRID, IGX, IGY, TIME, LEVEL, IVCORD, PARMU, PARMV,
          IRET )
```

Input parameters:

GDATTM	CHAR*	Input date/time
GLEVEL	CHAR*	Input level
GVCORD	CHAR*	Input vertical coordinate
GVECT	CHAR*	Diagnostic function

Output parameters:

PFUNC	CHAR*	Diagnostic error string
UGRID (IGX,IGY)	REAL	Output u component grid
VGRID (IGX,IGY)	REAL	Output v component grid
IGX	INTEGER	Number of points in x dir
IGY	INTEGER	Number of points in y dir
TIME (2)	CHAR*	Output date/time
LEVEL (2)	INTEGER	Output level
IVCORD	INTEGER	Output vertical coordinate
PARMU	CHAR*	Parameter name for u component
PARMV	CHAR*	Parameter name for v component
IRET	INTEGER	Return code 3 = user typed EXIT 0 = normal return -3 = parsing table is empty -5 = output grid not a vector -6 = wrong number of operands -7 = grid cannot be found -8 = grid is the wrong size -9 = incorrect operands -10 = internal grid list is full -11 = operand must be a vector -12 = operand must be a scalar -13 = operand must be from file -14 = DG_INIT not initialized -15 = polar grid cent. not valid -16 = map proj is invalid -17 = LEVEL must be a layer -18 = TIME must be a range -19 = invalid operator -20 = stack is full -21 = stack is empty -22 = TIME is invalid

-23 = LEVEL is invalid
-24 = IVCORD is invalid
-26 = layer of layers is invalid
-27 = time range layer invalid

2.8 DG_VECT - COMPUTE VECTOR GRID

This subroutine computes a grid diagnostic vector quantity. The u and v components returned in UGRID and VGRID are in north relative coordinates. GDATTM, GLEVEL, GVCORD and GVECT should have the values entered by the user.

```
DG_VECT  ( GDATTM, GLEVEL, GVCORD, GVECT, PFUNC, UGRID,
           VGRID, IGX, IGY, TIME, LEVEL, IVCORD, PARMU, PARMV,
           IRET )
```

Input parameters:

GDATTM	CHAR*	Input date/time
GLEVEL	CHAR*	Input level
GVCORD	CHAR*	Input vertical coordinate
GVECT	CHAR*	Diagnostic function

Output parameters:

PFUNC	CHAR*	Diagnostic error string
UGRID (IGX,IGY)	REAL	Output u component grid
VGRID (IGX,IGY)	REAL	Output v component grid
IGX	INTEGER	Number of points in x dir
IGY	INTEGER	Number of points in y dir
TIME (2)	CHAR*	Output date/time
LEVEL (2)	INTEGER	Output level
IVCORD	INTEGER	Output vertical coordinate
PARMU	CHAR*	Parameter name for u component
PARMV	CHAR*	Parameter name for v component
IRET	INTEGER	Return code 3 = user typed EXIT 0 = normal return -3 = GFUNC is blank -5 = output grid not a vector -6 = wrong number of operands -7 = grid cannot be found -8 = grid is the wrong size -9 = incorrect operands -10 = internal grid list is full -11 = operand must be a vector -12 = operand must be a scalar -13 = operand not in grid file -14 = DG_INIT not initialized -15 = polar grid center invalid -16 = map proj is invalid -17 = LEVEL must be a layer -18 = TIME must be a range -19 = invalid operator -20 = stack is full -21 = stack is empty -22 = TIME is invalid

-23 = LEVEL is invalid
-24 = IVCORD is invalid
-26 = layer of layers invalid
-27 = time range layer invalid

CHAPTER 3

DATA MANAGEMENT (DM) LIBRARY

DM_BEGS	Reset search
DM_CHNG	Change file access
DM_CLOS	Close a DM file
DM_CRET	Create a DM file
DM_CSRC	Set conditional search
DM_DALL	Delete by row or column
DM_DCLH	Delete a column header
DM_DCSR	Delete conditional search
DM_DDAT	Delete data
DM_DPSR	Delete primary search
DM_DRWH	Delete a row header
DM_FKEY	Determine key location
DM_FWRT	Flush write buffers
DM_GTIM	Return list of times
DM_KEYS	Return row and column keys
DM_LSTN	Locate station identifier
DM_LTIM	Locate date/time
DM_NEXT	Find next row and column
DM_OPEN	Open a DM file
DM_PART	Return part information
DM_PNAM	Return part names
DM_PSRC	Set primary search
DM_QDAT	Check for data
DM_RCLH	Read a column header
DM_RDTC	Read character data
DM_RDTI	Read integer data
DM_RDTR	Read real data
DM_RFHC	Read character file header
DM_RFHI	Read integer file header
DM_RFHR	Read real file header
DM_RRWH	Read a row header
DM_SRCH	Find particular row or column
DM_WCLH	Write a column header
DM_WDTC	Write character data
DM_WDTI	Write integer data

DM_WDTR	Write real data
DM_WFHC	Write character file header
DM_WFHI	Write integer file header
DM_WFHR	Write real file header
DM_WRWH	Write a row header

Data Management (DM) Library Summary

The data management library is the support library for reading and writing all GEMPAK files. In general, libraries specific to the various data types (SF, SN, GD) should be used by the applications programmer. This documentation is provided to assist in writing these data-type-specific subroutines.

Each DM file has rows, columns and parts. Rows and columns are identified by sequential numbers. Each row and column has a header containing information about the entire row or column. The keywords defining this information are specified when the data set is created and may be obtained using DM_KEYS. Header information is always stored as an array of integer values. For station data, the rows typically contain the date/time, while the columns typically contain information about individual stations. Note that not all station data is stored in this way. For example, ship data is stored in a single row with time and station information combined in a single header.

Parts in a DM file are identified by name. For conventional upper-air data, the six reports (TTAA, TTBB, PPBB, TTCC, TTDD, PPDD) are stored as six parts.

Data in a DM file are identified by a row number, column number and part name. If data are to be packed, the packing information must be provided when the file is created. The data will be packed and unpacked within the DM library, so the programmer can access the data as real values using subroutines DM_RDTR and DM_WDTR.

Information about the entire file may be stored in file headers. This information is stored using DM_WFHx and returned using DM_RFHx.

Subroutines to search for row and column headers meeting certain criteria are also available. DM_PSRC is used to define a primary search. The conditions for this search must always be met. In addition, conditional searches may be defined using DM_CSRC. These conditional searches may be additive or subtractive, meaning that rows/columns meeting the criteria will be added or subtracted from the list of valid rows/columns. When using these subroutines, DM_NEXT will return the numbers of the next row and column meeting the search criteria. The applications programmer should use the data-specific libraries and the location (LC) library to search DM data sets. DM_SRCH provides a simple search whose return code can be used to determine if the search criteria are ever met.

The subroutines DM_LSTN, DM_LTIM and DM_GTIM are provided to simplify access to DM files by data-specific libraries.

ERROR MESSAGES:

[DM -1] File ... could not be created.
[DM -2] File ... could not be opened.
[DM -3] Too many files open.
[DM -4] File is not open.
[DM -5] Invalid dimension sizes.
[DM -6] Write error.
[DM -7] Read error.
[DM -8] Undefined file header.
[DM -9] Invalid row or column location.
[DM -10] Invalid part name.
[DM -11] Undefined row or column header.
[DM -12] No more row/column headers.
[DM -13] No write access.
[DM -14] Invalid key name.
[DM -15] Data not available.
[DM -16] Invalid data packing terms.
[DM -17] Search criteria not met.
[DM -18] File header length too large.
[DM -19] Error packing/unpacking data.
[DM -20] File is not a GEMPAK DM file.
[DM -21] Incorrect data type.
[DM -22] Too many searches defined.
[DM -23] File was created on a different machine.
[DM -24] Invalid number of words to pack/unpack.
[DM -25] Invalid station keywords.
[DM -26] Invalid delete conditions.
[DM -27] Invalid time keywords.
[DM -28] Too many times.
[DM -29] Invalid file header name.
[DM -30] Close and reopen failed.
[DM -31] Error packing or unpacking grid.

DATA MANAGEMENT (DM) LIBRARY

DM Library Calls

DM_BEGS (iflno, / iret)

DM_CHNG (iflno, wrtflg, shrflg, / iret)

DM_CLOS (iflno, / iret)

DM_CRET (filnam, iftype, ifsrce, nfhdrs, fhdnam, ifhlen, ifhtyp, nrow, nrkeys, keyrow, ncol, nckeys, keycol, nprt, prtnam, lenhdr, ityprt, nparms, maxprm, prmnam, iscale, ioffst, nbits, / iflno, iret)

DM_CSRC (iflno, addsrc, nkeys, keynam, iloval, ihival, / iret)

DM_DALL (iflno, nkeys, keynam, iloval, ihival, / iret)

DM_DCLH (iflno, ipos, / iret)

DM_DCSR (iflno, / iret)

DM_DDAT (iflno, irow, icol, part, / iret)

DM_DPSR (iflno, / iret)

DM_DRWH (iflno, ipos, / iret)

DM_FKEY (iflno, keynam, / type, loc, iret)

DM_FWRT (iflno, / iret)

DM_GTIM (iflno, maxtim, / ntime, timlst, iret)

DM_KEYS (iflno, / nrkeys, keyrow, nckeys, keycol, iret)

DM_LSTN (iflno, / sttype, ilstid, ilstnm, ilslat, ilslon, ilselv, ilstat, ilcoun, iret)

DM_LTIM (iflno, / dttype, ildate, iltime, iret)

DM_NEXT (iflno, / irow, icol, iret)

DM_OPEN (filnam, wrtflg, shrflg, / iflno, iftype, ifsrce, nrow, ncol, nprt, nfhdrs, iret)

DM_PART (iflno, prtnam, / lenhdr, ityprt, nparms, prmnam, iscale, ioffst, nbits, iret)

DM_PNAM (iflno, / nprt, prtnam, iret)

DM_PSRC (iflno, nkeys, keynam, iloval, ihival, / iret)

```
DM_QDAT  ( iflno, irow, icol, part, / datflg, iret )
DM_RCLH  ( iflno, ipos, / iheadr, iret )
DM_RDTC  ( iflno, irow, icol, part, / idthdr, cdata, nchar, iret )
DM_RDTI  ( iflno, irow, icol, part, / idthdr, idata, nword, iret )
DM_RDTR  ( iflno, irow, icol, part, / idthdr, rdata, nword, iret )
DM_RFHC  ( iflno, fhdnam, mxchar, / cheadr, nchar, iret )
DM_RFHI  ( iflno, fhdnam, mxword, / iheadr, nword, iret )
DM_RFHR  ( iflno, fhdnam, mxword, / rheadr, nword, iret )
DM_RRWH  ( iflno, ipos, / iheadr, iret )
DM_SRCH  ( iflno, type, nkey, keyloc, keyval, / irwcl, iret )
DM_WCLH  ( iflno, ipos, iheadr, / jpos, iret )
DM_WDTC  ( iflno, irow, icol, part, idthdr, cdata, nchar, / iret )
DM_WDTI  ( iflno, irow, icol, part, idthdr, idata, nword, / iret )
DM_WDTR  ( iflno, irow, icol, part, idthdr, rdata, nword, / iret )
DM_WFHC  ( iflno, fhdnam, cheadr, nchar, / iret )
DM_WFHI  ( iflno, fhdnam, iheadr, nword, / iret )
DM_WFHR  ( iflno, fhdnam, rheadr, nword, / iret )
DM_WRWH  ( iflno, ipos, iheadr, / jpos, iret )
```

3.1 DM_BEGS - RESET SEARCH

This subroutine restarts a search at the beginning of a DM file.

DM_BEGS (IFLNO, IRET)

Input parameters:		
IFLNO	INTEGER	File number
Output parameters:		
IRET	INTEGER	Return code 0 = normal return -4 = file not open

3.2 DM_CHNG - CHANGE FILE ACCESS

This subroutine changes the access permissions for a DM file. If necessary, the file is closed and reopened with the requested access.

DM_CHNG (IFLNO, WRTFLG, SHRFLG, IRET)

Input parameters:

IFLNO	INTEGER	File number
WRTFLG	LOGICAL	Write flag
SHRFLG	LOGICAL	Shared access flag

Output parameters:

IRET	INTEGER	Return code 0 = normal return -30 = close & open failed

3.3 DM_CLOS - CLOSE A DM FILE

This subroutine closes a DM file and deallocates the file number.

DM_CLOS (IFLNO, IRET)

Input parameters:

IFLNO	INTEGER	File number

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file is not open -6 = write error

3.4 DM_CRET - CREATE A DM FILE

This subroutine creates a new DM file. The arrays PRMNAM, ISCALE, IOFFST, and NBITS must be two-dimensional arrays in the calling program whose first dimension is MAXPRM. After the file is created, it is left open with write access.

```
DM_CRET ( FILNAM, IFTYPE, IFSRCE, NFHDRS, FHDNAM, IFHLEN, IFHTYP,
          NROW,   NRKEYS, KEYROW, NCOL,   NCKEYS, KEYCOL, NPRT,
          PRTNAM, LENHDR, ITYPRT, NPARMS, MAXPRM, PRMNAM, ISCALE,
          IOFFST, NBITS,  IFLNO,  IRET )
```

Input parameters:

FILNAM	CHAR*	File name
IFTYPE	INTEGER	File type
IFSRCE	INTEGER	File source
NFHDRS	INTEGER	Number of file headers
FHDNAM (NFHDRS)	CHAR*4	File header names
IFHLEN (NFHDRS)	INTEGER	File header lengths
IFHTYP (NFHDRS)	INTEGER	File header data types
NROW	INTEGER	Number of rows
NRKEYS	INTEGER	Number of row keys
KEYROW (NRKEYS)	CHAR*4	Row key names
NCOL	INTEGER	Number of columns
NCKEYS	INTEGER	Number of column keys
KEYCOL (NCKEYS)	CHAR*4	Column key names
NPRT	INTEGER	Number of parts
PRTNAM (NPRT)	CHAR*4	Part names
LENHDR (NPRT)	INTEGER	Part header lengths
ITYPRT (NPRT)	INTEGER	Part data types
NPARMS (NPRT)	INTEGER	Number of parameters / part
MAXPRM	INTEGER	Maximum number of parameters
PRMNAM (MAXPRM,NPRT)	CHAR*	Parameter names
ISCALE (MAXPRM,NPRT)	INTEGER	Scaling for packed real
IOFFST (MAXPRM,NPRT)	INTEGER	Offset for packed real
NBITS (MAXPRM,NPRT)	INTEGER	Number of bits for packed real

Output parameters:

IFLNO	INTEGER	File number
IRET	INTEGER	Return code 0 = normal return -1 = file cannot be created -3 = too many files open -5 = invalid dimension sizes -6 = write error -16 = invalid packing terms

3.5 DM_CSRC - SET CONDITIONAL SEARCH

This subroutine defines criteria for a conditional search. The conditional search will be made if the primary search succeeds.

DM_CSRC (IFLNO, ADDSRC, NKEYS, KEYNAM, ILOVAL, IHIVAL, IRET)

Input parameters:

IFLNO	INTEGER	File number
ADDSRC	LOGICAL	Additive search flag
NKEYS	INTEGER	Number of keys used in search
KEYNAM (NKEYS)	CHAR*4	Key names
ILOVAL (NKEYS)	INTEGER	Low values
IHIVAL (NKEYS)	INTEGER	High values

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -14 = invalid key name -22 = too many searches

3.6 DM_DALL - DELETE BY ROW OR COLUMN

This subroutine deletes data for all locations which match the given search criteria. Data for all parts are deleted along with the appropriate headers. This subroutine packs the data into large free blocks and is preferred to deleting single parts using DM_DDAT.

DM_DALL (IFLNO, NKEYS, KEYNAM, ILOVAL, IHIVAL, IRET)

Input parameters:

IFLNO	INTEGER	File number
NKEYS	INTEGER	Number of keys in search
KEYNAM (NKEYS)	CHAR*4	Key names
ILOVAL (NKEYS)	INTEGER	Minimum values
IHIVAL (NKEYS)	INTEGER	Maximum values

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -13 = no write access -17 = search criteria not met -26 = invalid delete conditions

3.7 DM_DCLH - DELETE A COLUMN HEADER

This subroutine deletes a column header from a DM file.

DM_DCLH (IFLNO, IPOS, IRET)

Input parameters:

IFLNO	INTEGER	File number
IPOS	INTEGER	Location

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file is not open -6 = write error -9 = invalid column -13 = no write access

3.8 DM_DCSR - DELETE CONDITIONAL SEARCH

This subroutine deletes the conditional searches for a DM file.

DM_DCSR (IFLNO, IRET)

Input parameters:

IFLNO	INTEGER	File number

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open

3.9 DM_DDAT - DELETE DATA

This subroutine deletes data for a single row, column and part from a DM file. If an entire column or row is to be deleted, the subroutine DM_DALL should be used.

DM_DDAT (IFLNO, IROW, ICOL, PART, IRET)

Input parameters:

IFLNO	INTEGER	File number
IROW	INTEGER	Row number
ICOL	INTEGER	Column number
PART	CHAR*4	Part name

Output parameters:

IRET	INTEGER	Return code
		0 = normal return
		-4 = file not open
		-6 = write error
		-7 = read error
		-9 = invalid row/column
		-10 = invalid part name
		-13 = no write access

3.10 DM_DPSR - DELETE PRIMARY SEARCH

This subroutine deletes the primary search for a DM file.

DM_DPSR (IFLNO, IRET)

Input parameters:

IFLNO	INTEGER	File number

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open

3.11 DM_DRWH - DELETE A ROW HEADER

This subroutine deletes a row header from a DM file.

DM_DRWH (IFLNO, IPOS, IRET)

Input parameters:

IFLNO	INTEGER	File number
IPOS	INTEGER	Location

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file is not open -6 = write error -9 = invalid location -13 = no write access

3.12 DM_FKEY - DETERMINE KEY LOCATION

This subroutine finds the type and location of a row or column key. If the key is not found, the location is set to 0.

DM_FKEY (IFLNO, KEYNAM, TYPE, LOC, IRET)

Input parameters:

IFLNO	INTEGER	File number
KEYNAM	CHAR*4	Key name

Output parameters:

TYPE	CHAR*	Type : ROW or COL
LOC	INTEGER	Key location
IRET	INTEGER	Return code 0 = normal return -4 = file is not open -14 = invalid key name

3.13 DM_FWRT - FLUSH WRITE BUFFERS

This subroutine flushes the write buffers for a DM file.

DM_FWRT (IFLNO, IRET)

Input parameters:

IFLNO	INTEGER	File number

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error

3.14 DM_GTIM - RETURN LIST OF TIMES

This subroutine returns a list of the GEMPAK date/times found in the file. The times are sorted from earliest to latest.

DM_GTIM (IFLNO, MAXTIM, NTIME, TIMLST, IRET)

Input parameters:

IFLNO	INTEGER	File number
MAXTIM	INTEGER	Max number of times to return

Output parameters:

NTIME	INTEGER	Number of times returned
TIMLST (NTIME)	CHAR*15	List of times
IRET	INTEGER	Return code
		0 = normal return
		-4 = file not open
		-27 = invalid time keywords
		-28 = too many times

3.15 DM_KEYS - RETURN ROW AND COLUMN KEYS

This subroutine returns the row and column keys in a DM file.

DM_KEYS (IFLNO, NRKEYS, KEYROW, NCKEYS, KEYCOL, IRET)

Input parameters:

IFLNO	INTEGER	File number

Output parameters:

NRKEYS	INTEGER	Number of row keys
KEYROW (NRKEYS)	CHAR*4	Row keys
NCKEYS	INTEGER	Number of column keys
KEYCOL (NCKEYS)	CHAR*4	Column keys
IRET	INTEGER	Return code 0 = normal return -4 = file is not open

3.16 DM_LSTN - LOCATE STATION IDENTIFIER

This subroutine finds the location of the station keywords. Both SLAT and SLON must be row or column keys. The locations of the keywords STID, STNM, SELV, STAT and COUN are also checked. If present, they must be the same type as the SLAT and SLON keys. If a key is not found, the location is set to 0.

```
DM_LSTN  ( IFLNO,  STTYPE, ILSTID, ILSTNM, ILSLAT, ILSLON,
           ILSELV, ILSTAT, ILCOUN, IRET )
```

Input parameters:

IFLNO	INTEGER	File number

Output parameters:

STTYPE	CHAR*	Location type: ROW or COL
ILSTID	INTEGER	Location of STID
ILSTNM	INTEGER	Location of STNM
ILSLAT	INTEGER	Location of SLAT
ILSLON	INTEGER	Location of SLON
ILSELV	INTEGER	Location of SELV
ILSTAT	INTEGER	Location of STAT
ILCOUN	INTEGER	Location of COUN
IRET	INTEGER	Return code 0 = normal return -4 = file not open -25 = invalid station keywords

3.17 DM_LTIM - LOCATE DATE/TIME

This subroutine finds the location of the DATE and TIME keywords in a DM file. Both keys must be row keys or column keys.

DM_LTIM (IFLNO, DTTYPE, ILDATE, ILTIME, IRET)

Input parameters:

IFLNO	INTEGER	File number

Output parameters:

DTTYPE	CHAR*	Location type: ROW or COL
ILDATE	INTEGER	Location of DATE
ILTIME	INTEGER	Location of TIME
IRET	INTEGER	Return code 0 = normal return -4 = file not open -27 = invalid time keywords

3.18 DM_NEXT - FIND NEXT ROW AND COLUMN

This subroutine returns the location of the next row and column meeting the search criteria.

DM_NEXT (IFLNO, IROW, ICOL, IRET)

Input parameters:

IFLNO	INTEGER	File number

Output parameters:

IROW	INTEGER	Row number
ICOL	INTEGER	Column number
IRET	INTEGER	Return code 0 = normal return -4 = file not open -17 = search criteria not met

3.19 DM_OPEN - OPEN A DM FILE

This subroutine opens a data management (DM) file.

```
DM_OPEN ( FILNAM, WRTFLG, SHRFLG, IFLNO, IFTYPE, IFSRCE, NROW,
          NCOL,   NPRT,   NFHDRS, IRET )
```

Input parameters:

FILNAM	CHAR*	File name
WRTFLG	LOGICAL	Write access flag
SHRFLG	LOGICAL	Shared file flag

Output parameters:

IFLNO	INTEGER	File number
IFTYPE	INTEGER	File type
IFSRCE	INTEGER	File source
NROW	INTEGER	Number of rows
NCOL	INTEGER	Number of columns
NPRT	INTEGER	Number of parts
NFHDRS	INTEGER	Number of file headers
IRET	INTEGER	Return code 0 = normal return -2 = file cannot be opened -3 = too many files open -6 = write error -7 = read error -23 = wrong machine type -32 = invalid machine for write

3.20 DM_PART - RETURN PART INFORMATION

This subroutine returns information for a specific part.

```
DM_PART  ( IFLNO, PRTNAM, LENHDR, ITYPRT, NPARMS, PRMNAM, ISCALE,
           IOFFST, NBITS, IRET )
```

Input parameters:

IFLNO	INTEGER	File number
PRTNAM	CHAR*4	Part name

Output parameters:

LENHDR	INTEGER	Length of data header
ITYPRT	INTEGER	Data type
NPARMS	INTEGER	Number of parameters
PRMNAM (NPARMS)	CHAR*4	Parameter names
ISCALE (NPARMS)	INTEGER	Scaling term
IOFFST (NPARMS)	INTEGER	Offset
NBITS (NPARMS)	INTEGER	Number of bits
IRET	INTEGER	Return code 0 = normal return -4 = file not open -10 = invalid part name

3.21 DM_PNAM - RETURN PART NAMES

This subroutine returns the names of all the parts in a DM file.

DM_PNAM (IFLNO, NPRT, PRTNAM, IRET)

Input parameters:

IFLNO	INTEGER	File number

Output parameters:

NPRT	INTEGER	Number of parts
PRTNAM (NPRT)	CHAR*4	Part names
IRET	INTEGER	Return code 0 = normal return -4 = file not open

3.22 DM_PSRC - SET PRIMARY SEARCH

This subroutine defines criteria for the primary search. If the result of this primary search is false for any location, no conditional search will be made.

DM_PSRC (IFLNO, NKEYS, KEYNAM, ILOVAL, IHIVAL, IRET)

Input parameters:

IFLNO	INTEGER	File number
NKEYS	INTEGER	Number of keys used in search
KEYNAM (NKEYS)	CHAR*4	Key names
ILOVAL (NKEYS)	INTEGER	Low values
IHIVAL (NKEYS)	INTEGER	High values

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -14 = invalid key name

3.23 DM_QDAT - CHECK FOR DATA

This subroutine sets a flag indicating whether data for a given row, column and part is stored in a DM file.

DM_QDAT (IFLNO, IROW, ICOL, PART, DATFLG, IRET)

Input parameters:

IFLNO	INTEGER	File number
IROW	INTEGER	Row number
ICOL	INTEGER	Column number
PART	CHAR*4	Part name

Output parameters:

DATFLG	LOGICAL	Data present flag
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -7 = read error -9 = invalid location -10 = invalid part name

3.24 DM_RCLH - READ A COLUMN HEADER

This subroutine reads a column header from a DM file.

DM_RCLH (IFLNO, IPOS, IHEADR, IRET)

Input parameters:

IFLNO	INTEGER	File number
IPOS	INTEGER	Location

Output parameters:

IHEADR (*)	INTEGER	Header array
IRET	INTEGER	Return code 0 = normal return -4 = file is not open -9 = invalid column -11 = undefined header

3.25 DM_RDTC - READ CHARACTER DATA

This subroutine reads character data from a DM file.

DM_RDTC (IFLNO, IROW, ICOL, PART, IDTHDR, CDATA, NCHAR, IRET)

Input parameters:

IFLNO	INTEGER	File number
IROW	INTEGER	Row number
ICOL	INTEGER	Column number
PART	CHAR*4	Part name

Output parameters:

IDTHDR (*)	INTEGER	Data header
CDATA	CHAR*NCHAR	Data
NCHAR	INTEGER	Length of string
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -7 = read error -9 = invalid location -10 = invalid part name -15 = data not available -21 = incorrect data type

3.26 DM_RDTI - READ INTEGER DATA

This subroutine reads integer data from a DM file.

DM_RDTI (IFLNO, IROW, ICOL, PART, IDTHDR, IDATA, NWORD, IRET)

Input parameters:

IFLNO	INTEGER	File number
IROW	INTEGER	Row number
ICOL	INTEGER	Column number
PART	CHAR*4	Part name

Output parameters:

IDTHDR (*)	INTEGER	Data header
IDATA (NWORD)	INTEGER	Data
NWORD	INTEGER	Length of data array
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -7 = read error -9 = invalid location -10 = invalid part name -15 = data not available -21 = incorrect data type

3.27 DM_RDTR - READ REAL DATA

This subroutine reads real data from a DM file.

DM_RDTR (IFLNO, IROW, ICOL, PART, IDTHDR, RDATA, NWORD, IRET)

Input parameters:

IFLNO	INTEGER	File number
IROW	INTEGER	Row number
ICOL	INTEGER	Column number
PART	CHAR*4	Part name

Output parameters:

IDTHDR (*)	INTEGER	Data header
RDATA (NWORD)	REAL	Data
NWORD	INTEGER	Length of data array
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -7 = read error -9 = invalid location -10 = invalid part name -15 = data not available -21 = incorrect data type

3.28 DM_RFHC - READ CHARACTER FILE HEADER

This subroutine reads a character file header from a DM file. The length of the file header must be less than MXCHAR.

DM_RFHC (IFLNO, FHDNAM, MXCHAR, CHEADR, NCHAR, IRET)

Input parameters:

IFLNO	INTEGER	File number
FHDNAM	CHAR*4	File header name
MXCHAR	INTEGER	Maximum characters to return

Output parameters:

CHEADR	CHAR*NCHAR	File header
NCHAR	INTEGER	Header length
IRET	INTEGER	Return code
		0 = normal return
		-4 = file not open
		-6 = write error
		-7 = read error
		-8 = undefined file header
		-18 = file header too long
		-21 = incorrect data type
		-29 = invalid file hdr name

3.29 DM_RFHI - READ INTEGER FILE HEADER

This subroutine reads an integer file header from a DM file. The length of the file header must be less than MXWORD.

DM_RFHI (IFLNO, FHDNAM, MXWORD, IHEADR, NWORD, IRET)

Input parameters:

IFLNO	INTEGER	File number
FHDNAM	CHAR*4	File header name
MXWORD	INTEGER	Maximum words to return

Output parameters:

IHEADR (NWORD)	INTEGER	File header
NWORD	INTEGER	Header length
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -7 = read error -8 = undefined file header -18 = file header too long -21 = incorrect data type -29 = invalid file hdr name

3.30 DM_RFHR - READ REAL FILE HEADER

This subroutine reads a real file header from a DM file. The length of the file header must be less than MXWORD.

DM_RFHR (IFLNO, FHDNAM, MXWORD, RHEADR, NWORD, IRET)

Input parameters:

IFLNO	INTEGER	File number
FHDNAM	CHAR*4	File header name
MXWORD	INTEGER	Maximum words to return

Output parameters:

RHEADR (NWORD)	REAL	File header
NWORD	INTEGER	Header length
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -7 = read error -8 = file header undefined -18 = file header too long -21 = incorrect data type -29 = invalid file hdr name

3.31 DM_RRWH - READ A ROW HEADER

This subroutine reads a row header from a DM file.

DM_RRWH (IFLNO, IPOS, IHEADR, IRET)

Input parameters:

IFLNO	INTEGER	File number
IPOS	INTEGER	Location

Output parameters:

IHEADR (*)	INTEGER	Header array
IRET	INTEGER	Return code
		0 = normal return
		-4 = file is not open
		-9 = invalid row
		-11 = undefined header

3.32 DM_SRCH - FIND PARTICULAR ROW OR COLUMN

This subroutine searches a DM file for rows or columns which match the given input values.

DM_SRCH (IFLNO, TYPE, NKEY, KEYLOC, KEYVAL, IRWCL, IRET)

Input parameters:

IFLNO	INTEGER	File number
TYPE	CHAR*	Dimension type : ROW or COL
NKEY	INTEGER	Number of keys to search
KEYLOC (NKEY)	INTEGER	Key locations
KEYVAL (NKEY)	INTEGER	Key values

Output parameters:

IRWCL	INTEGER	Search location
IRET	INTEGER	Return code 0 = normal return -4 = file is not open -17 = search criteria not met

3.33 DM_WCLH - WRITE A COLUMN HEADER

This subroutine writes a column header to a DM file. If the value for IPOS is 0, the next available location will be used. The variables contained in the row headers can be determined using DM_KEYS.

DM_WCLH (IFLNO, IPOS, IHEADR, JPOS, IRET)

Input parameters:

IFLNO	INTEGER	File number
IPOS	INTEGER	Location
IHEADR (*)	INTEGER	Header array

Output parameters:

JPOS	INTEGER	Actual header location
IRET	INTEGER	Return code
		0 = normal return
		-4 = file is not open
		-6 = write error
		-9 = invalid location
		-12 = no more column headers
		-13 = no write access

3.34 DM_WDTC - WRITE CHARACTER DATA

This subroutine writes character data to a DM file.

DM_WDTC (IFLNO, IROW, ICOL, PART, IDTHDR, CDATA, NCHAR, IRET)

Input parameters:

IFLNO	INTEGER	File number
IROW	INTEGER	Row number
ICOL	INTEGER	Column number
PART	CHAR*4	Part name
IDTHDR (*)	INTEGER	Data header
CDATA	CHAR*NCHAR	Data
NCHAR	INTEGER	Length of string

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -9 = invalid row or column -10 = invalid part name -13 = no write access -21 = incorrect data type

3.35 DM_WDTI - WRITE INTEGER DATA

This subroutine writes integer data to a DM file.

DM_WDTI (IFLNO, IROW, ICOL, PART, IDTHDR, IDATA, NWORD, IRET)

Input parameters:

IFLNO	INTEGER	File number
IROW	INTEGER	Row number
ICOL	INTEGER	Column number
PART	CHAR*4	Part name
IDTHDR (*)	INTEGER	Data header
IDATA (NWORD)	INTEGER	Data
NWORD	INTEGER	Length of data array

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -9 = invalid row or column -10 = invalid part name -13 = no write access -21 = incorrect data type

3.36 DM_WDTR - WRITE REAL DATA

This subroutine writes real data to a DM file.

DM_WDTR (IFLNO, IROW, ICOL, PART, IDTHDR, RDATA, NWORD, IRET)

Input parameters:

IFLNO	INTEGER	File number
IROW	INTEGER	Row number
ICOL	INTEGER	Column number
PART	CHAR*4	Part name
IDTHDR (*)	INTEGER	Data header
RDATA (NWORD)	REAL	Data
NWORD	INTEGER	Length of data array

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -9 = invalid row or column -10 = invalid part name -13 = no write access -21 = incorrect data type

3.37 DM_WFHC - WRITE CHARACTER FILE HEADER

This subroutine writes a character file header to a DM file. The length of the file header must be less than the length given when the file was created. When the file header is read, the length input in this subroutine will be returned.

DM_WFHC (IFLNO, FHDNAM, CHEADR, NCHAR, IRET)

Input parameters:

IFLNO	INTEGER	File number
FHDNAM	CHAR*4	File header name
CHEADR	CHAR*NCHAR	File header
NCHAR	INTEGER	Header length

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -7 = read error -13 = no write access -18 = file header too long -21 = incorrect data type -29 = invalid file hdr name

3.38 DM_WFHI - WRITE INTEGER FILE HEADER

This subroutine writes an integer file header to a DM file. The length of the file header must be less than the length given when the file was created. When the file header is read, the length input in this subroutine will be returned.

DM_WFHI (IFLNO, FHDNAM, IHEADR, NWORD, IRET)

Input parameters:

IFLNO	INTEGER	File number
FHDNAM	CHAR*4	File header name
IHEADR (NWORD)	INTEGER	File header
NWORD	INTEGER	Header length

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -7 = read error -13 = no write access -18 = file header too long -21 = incorrect data type -29 = invalid file hdr name

3.39 DM_WFHR - WRITE REAL FILE HEADER

This subroutine writes a real-valued file header to a DM file. The length of the file header must be less than the length given when the file was created. When the file header is read, the length input in this subroutine will be returned.

DM_WFHR (IFLNO, FHDNAM, RHEADR, NWORD, IRET)

Input parameters:

IFLNO	INTEGER	File number
FHDNAM	CHAR*4	File header name
RHEADR (NWORD)	REAL	File header
NWORD	INTEGER	Header length

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = write error -7 = read error -13 = no write access -18 = file header too long -21 = incorrect data type -29 = invalid file hdr name

3.40 DM_WRWH - WRITE A ROW HEADER

This subroutine writes a row header to a DM file. If the value for IPOS is 0, the next available location will be used. The variables contained in the row headers can be determined using DM_KEYS.

DM_WRWH (IFLNO, IPOS, IHEADR, JPOS, IRET)

Input parameters:

IFLNO	INTEGER	File number
IPOS	INTEGER	Location
IHEADR (*)	INTEGER	Header array

Output parameters:

JPOS	INTEGER	Actual header location
IRET	INTEGER	Return code 0 = normal return -4 = file is not open -6 = write error -9 = invalid row -12 = no more row headers -13 = no write access

CHAPTER 4

DATA PACKING (DP) LIBRARY

DP_ENDP	Release packing number
DP_FILE	Read packing file
DP_PACK	Pack data
DP_PDEC	Pack grid in DEC format
DP_PDIF	Pack grid in DIF format
DP_PGRB	Pack grid in GRIB format
DP_SETP	Define packing terms
DP_TERM	Compute packing terms
DP_UDIF	Unpack grid in DIF format
DP_UGRB	Unpack grid in GRIB format
DP_UNMC	Unpack grid in NMC format
DP_UNPK	Unpack data

Data Packing (DP) Library Summary

The data packing library provides subroutines for packing real data values into a bit string and for unpacking these data. The bit string is stored and retrieved as an integer data array. In general, packing and unpacking is done in the DM library subroutines and is the responsibility of the programmer.

Station data is packed using DP_PACK and unpacked using DP_UNPK. The DP_PACK subroutine packs a real data value by applying a scale factor and an offset which transforms the expected range of data values into a small integer range. The following equation is used:

IPACK = NINT (DATA / SCALE) - IOFFST

The scale factor, SCALE, is 10 ** LOGSCL where LOGSCL is specified in DP_SETP.

LOGSCL, IOFFST, and NBITS must be defined by a call to DP_SETP before any packing or unpacking is done. The scale factor is specified in terms of its base-10 logarithm. These terms may be determined from the range and resolution desired using the subroutine DP_TERM.

Once DP_SETP has been called to define the packing parameters, either DP_PACK or DP_UNPK may be called repeatedly for data records to be packed or unpacked. The DP library allows multiple definitions. Each definition is identified by a packing number.

There are several packing schemes available for gridded data. These are called the GEMPAK GRIB, DIF and NMC formats.

The GEMPAK GRIB format is similar to the WMO GRIB format except that missing data points may be stored and retrieved and the scaling factor need not be a power of 2. The equations used are:

IDATA = NINT ((GRID - QMIN) / SCALE)
GRID = QMIN + IDATA * SCALE

DP_PGRB may be used to pack the data. In this case, SCALE will be a power of 2 and missing data may be stored and retrieved. DP_PDEC will pack data in a minimum number of bits to retain the requested decimal precision. SCALE will not necessarily be a power of 2. Data stored by either subroutine or as received in GRIB format may be unpacked using DP_UGRB.

The GEMPAK DIF format computes the difference between points along a row of data. At the first point in a row, the difference from the first point in the previous row is used. These differences are used

in the equations:

```
IDIF  =  NINT ( ( GDIF - DIFMIN ) / SCALE )
GDIF  =  DIFMIN + IDIF * SCALE
```

These grids may be packed using DP_PDIF and unpacked using DP_UDIF.

NMC 16-bit grids are saved directly from NMC. They may be unpacked using DP_UNMC which uses the equation:

```
GRID  = AVG + IDATA * SCALE
```

ERROR MESSAGES:

```
[DP  -1]  Packing terms are not defined.
[DP  -2]  Too many packing sets are defined.
[DP  -3]  Invalid number of data values.
[DP  -4]  Invalid number of bits specified.
[DP  -5]  Datamax less than datamin.
[DP  -6]  Invalid resolution.
[DP  -7]  Open error on parameter packing file.
[DP  -8]  Read error on parameter packing file.
[DP  -9]  Packed and unpacked data are mixed.
[DP -10]  NBITS is invalid for grid packing.
[DP -11]  The grid data has no range.
[DP -12]  SCALE is invalid for grid packing.
```

DP Library Calls

DP_ENDP	(ipkno, / iret)
DP_FILE	(prmfil, / nparm, parms, logscl, ioffst, nbits, pkflg, iret)
DP_PACK	(ipkno, data, / ibitst, iret)
DP_PDEC	(grid, igx, igy, ires, / idata, lendat, qmin, scale, nbits, iret)
DP_PDIF	(grid, igx, igy, nbits, / idata, lendat, p1, difmin, scale, iret)
DP_PGRB	(grid, igx, igy, nbits, / idata, lendat, qmin, scale, iret)
DP_SETP	(ndata, logscl, iofset, nbits, / ipkno, nwords, iret)
DP_TERM	(datmin, datmax, res, / logscl, iofset, nbits, iret)
DP_UDIF	(idata, kxky, nbits, p1, difmin, scale, misflg, kx, / grid, iret)
DP_UGRB	(idata, kxky, nbits, qmin, scale, misflg, / grid, iret)
DP_UNMC	(idata, kxky, nbits, ref, scale, misflg, / grid, iret)
DP_UNPK	(ipkno, ibitst, / data, iret)

4.1 DP_ENDP - RELEASE PACKING NUMBER

This subroutine releases a packing number for the DP library.

DP_ENDP (IPKNO, IRET)

Input parameters:

IPKNO	INTEGER	Packing number

Output parameters:

IRET	INTEGER	Return code 0 = normal return -1 = invalid packing number

4.2 DP_FILE - READ PACKING FILE

This subroutine reads a parameter-packing file. The parameters in the file and the data-packing terms are returned. If none of the data is to be packed, PKFLG is set to false. If some of the data is to be packed and some is not, an error is returned.

Parameter-packing file format:

Each parameter in the file must be described on a single line containing the following items separated by blanks or tabs:

parameter name	CHAR*4
minimum data value	REAL
maximum data value	REAL
resolution	REAL

The resolution should be an integral power of 10. If not, the next smaller resolution will be used (e.g., res = .5 will become .1). If the resolution is 0 or if the minimum, maximum and resolution are not present, the data will not be packed.

```
DP_FILE  ( PRMFIL, NPARM, PARMS, LOGSCL, IOFFST, NBITS, PKFLG,
           IRET )
```

Input parameters:

PRMFIL	CHAR*	Parameter packing file name

Output parameters:

NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*	Parameter names
LOGSCL (NPARM)	INTEGER	Log10 of scale factor
IOFFST (NPARM)	INTEGER	Offset
NBITS (NPARM)	INTEGER	Number of bits
PKFLG	LOGICAL	Packing flag
IRET	INTEGER	Return code 0 = normal return -3 = invalid number of parms -7 = packing file not opened -8 = file read error -9 = packed and unpacked data mixed

4.3 DP_PACK - PACK DATA

This subroutine packs an array of real values into a continuous bit string which is returned in the IBITST integer array. The subroutine DP_SETP must be called first to define the data packing terms.

DP_PACK (IPKNO, DATA, IBITST, IRET)

Input parameters:

IPKNO	INTEGER	Packing number
DATA (*)	REAL	Data values to be packed

Output parameters:

IBITST	INTEGER	Packed data
IRET	INTEGER	Return code 0 = normal return -1 = packing terms undefined

4.4 DP_PDEC - PACK GRID IN DEC FORMAT

This subroutine uses the precision specified to pack a grid into the GEMPAK GRIB format. The precision specifies the power of 10 to be used in scaling the data before converting to an integer. The minimum number of bits required to maintain the precision is computed. The GEMPAK GRIB packing and unpacking equations are:

```
IDATA  =  NINT ( ( GRID - QMIN ) / SCALE )
GRID   =  QMIN  +  IDATA * SCALE
```

```
DP_PDEC  ( GRID, IGX, IGY, IRES, IDATA, LENDAT, QMIN, SCALE, NBITS,
           IRET )
```

Input parameters:

GRID (IGX,IGY)	REAL	Grid data
IGX	INTEGER	Number of points in x dir
IGY	INTEGER	Number of points in y dir
IRES	INTEGER	Precision as power of 10

Output parameters:

IDATA (LENDAT)	INTEGER	Packed data
LENDAT	INTEGER	Length of packed data array
QMIN	REAL	Minimum value of grid
SCALE	REAL	Scaling factor
NBITS	INTEGER	Number of bits
IRET	INTEGER	Return code 0 = normal return -10 = NBITS invalid -11 = invalid data range

4.5 DP_PDIF - PACK GRID IN DIF FORMAT

This subroutine packs a grid into the GEMPAK DIF format. The value of the difference between a point and the previous point is packed using the equations:

```
IDIF = NINT ( ( GDIF - DIFMIN ) / SCALE )
GDIF = DIFMIN + IDIF * SCALE
```

```
DP_PDIF ( GRID, IGX, IGY, NBITS, IDATA, LENDAT, P1, DIFMIN,
          SCALE, IRET )
```

Input parameters:

GRID (IGX,IGY)	REAL	Grid data
IGX	INTEGER	Number of points in x dir
IGY	INTEGER	Number of points in y dir
NBITS	INTEGER	Number of bits

Output parameters:

IDATA (LENDAT)	INTEGER	Packed data
LENDAT	INTEGER	Length of packed data array
P1	REAL	Value of first grid point
DIFMIN	REAL	Minimum value of differences
SCALE	REAL	Scaling factor
IRET	INTEGER	Return code 0 = normal return -10 = NBITS invalid -11 = invalid data range

4.6 DP_PGRB - PACK GRID IN GRIB FORMAT

This subroutine packs a grid into the GEMPAK GRIB format using the number of bits specified. The packing and unpacking equations are:

```
IDATA  =  NINT ( ( GRID - QMIN ) / SCALE )
GRID   =  QMIN  +  IDATA * SCALE
```

```
DP_PGRB  ( GRID, IGX, IGY, NBITS, IDATA, LENDAT, QMIN, SCALE,
           IRET )
```

Input parameters:

GRID (IGX,IGY)	REAL	Grid data
IGX	INTEGER	Number of points in x dir
IGY	INTEGER	Number of points in y dir
NBITS	INTEGER	Number of bits

Output parameters:

IDATA (LENDAT)	INTEGER	Packed data
LENDAT	INTEGER	Length of packed data array
QMIN	REAL	Minimum value of grid
SCALE	REAL	Scaling factor
IRET	INTEGER	Return code 0 = normal return -10 = NBITS invalid -11 = invalid data range

4.7 DP_SETP - DEFINE PACKING TERMS

This subroutine defines the terms needed for data packing and unpacking. It must be called once for each set of data. Records may be packed or unpacked by calls to DP_PACK or DP_UNPK. The subroutine DP_TERM is provided for computing the values needed by this subroutine. LOGSCL is the power of 10 to be used in scaling data.

```
DP_SETP  ( NDATA, LOGSCL, IOFSET, NBITS, IPKNO, NWORDS, IRET )

Input parameters:
   NDATA              INTEGER          Number of data values
   LOGSCL  (NDATA)    INTEGER          Log10 of scale factor
   IOFSET  (NDATA)    INTEGER          Offset
   NBITS   (NDATA)    INTEGER          Number of bits

Output parameters:
   IPKNO              INTEGER          Packing number
   NWORDS             INTEGER          Number of words
   IRET               INTEGER          Return code
                                         0 = normal return
                                        -2 = no more packing numbers
                                        -3 = NDATA invalid
                                        -4 = NBITS invalid
```

4.8 DP_TERM - COMPUTE PACKING TERMS

This subroutine computes the terms required by the data-packing subroutines. The scale factor, offset, and number of bits are computed from the minimum, maximum and resolution for each data item. These terms are computed for a single item in this subroutine. Therefore, this subroutine must be called for each data item to be packed.

The resolution must be an integral power of 10. If not, the next smaller resolution will be used. For example: RES = .5 will use a resolution of .1 . LOGSCL is the base 10 logarithm of the value to be used in scaling data. NBITS must be less than 32.

DP_TERM (DATMIN, DATMAX, RES, LOGSCL, IOFSET, NBITS, IRET)

Input parameters:

DATMIN	REAL	Minimum data value
DATMAX	REAL	Maximum data value
RES	REAL	Resolution to be retained

Output parameters:

LOGSCL	INTEGER	Log10 of scaling factor
IOFSET	INTEGER	Data offset
NBITS	INTEGER	Number of bits
IRET	INTEGER	Return code 0 = normal return -5 = DATMAX less than DATMIN -6 = invalid resolution

4.9 DP_UDIF - UNPACK GRID IN DIF FORMAT

This subroutine unpacks a grid in the GEMPAK DIF format. The value of the difference between a point and the previous point is packed using the equations:

```
IDIF  =  NINT ( ( GDIF - DIFMIN ) / SCALE )
GDIF  =  DIFMIN  +  IDIF * SCALE
```

```
DP_UDIF  ( IDATA, KXKY, NBITS, P1, DIFMIN, SCALE, MISFLG, KX,
           GRID, IRET )
```

Input parameters:

IDATA (*)	INTEGER	Packed data
KXKY	INTEGER	Total number of grid points
NBITS	INTEGER	Number of bits
P1	REAL	Value of first grid point
DIFMIN	REAL	Minimum value of differences
SCALE	REAL	Scaling factor
MISFLG	LOGICAL	Missing data flag
KX	INTEGER	Number of points in x dir

Output parameters:

GRID (KXKY)	REAL	Grid data
IRET	INTEGER	Return code 0 = normal return -10 = NBITS invalid -11 = invalid data range

4.10 DP_UGRB - UNPACK GRID IN GRIB FORMAT

This subroutine unpacks a grid into the GEMPAK GRIB format. The packing and unpacking equations are:

```
IDATA  =  NINT ( ( GRID - QMIN ) / SCALE )
GRID   =  QMIN  +  IDATA * SCALE
```

DP_UGRB (IDATA, KXKY, NBITS, QMIN, SCALE, MISFLG, GRID, IRET)

Input parameters:

IDATA (*)	INTEGER	Packed data
KXKY	INTEGER	Number of grid points
NBITS	INTEGER	Number of bits
QMIN	REAL	Minimum value of grid
SCALE	REAL	Scaling factor
MISFLG	LOGICAL	Missing data flag

Output parameters:

GRID (KXKY)	REAL	Grid data
IRET	INTEGER	Return code 0 = normal return -10 = NBITS invalid -12 = invalid scale

4.11 DP_UNMC - UNPACK GRID IN NMC FORMAT

This subroutine unpacks a grid in the NMC format. The unpacking equation is:

GRID = REF + IDATA * SCALE

Each grid point must be packed into 16 bits which can be treated as an INTEGER*2 word. The scaling factor is a multiplier for the data. It must be set to 1 / 2 ** (15-N) where N is the exponent with the original NMC grid. This subroutine assumes there is no missing data.

DP_UNMC (IDATA, KXKY, NBITS, REF, SCALE, MISFLG, GRID, IRET)

Input parameters:

IDATA (KXKY)	INTEGER	Packed data
KXKY	INTEGER	Number of grid points
NBITS	INTEGER	Number of bits
REF	REAL	Reference value of grid
SCALE	REAL	Scaling factor
MISFLG	LOGICAL	Missing data flag

Output parameters:

GRID (KXKY)	REAL	Grid data
IRET	INTEGER	Return code 0 = normal return -10 = NBITS invalid -12 = invalid scale

4.12 DP_UNPK - UNPACK DATA

This subroutine unpacks a bit string from an integer array that was packed by the subroutine DP_PACK. The unpacked data is returned in a real array. DP_SETP must be called to define the packing terms before this subroutine is called.

DP_UPCK (IPKNO, BITSTR, DATA, IRET)

Input parameters:

IPKNO	INTEGER	Packing number
IBITST	INTEGER	Packed data array

Output parameters:

DATA (*)	REAL	Unpacked data values
IRET	INTEGER	Return code
		0 = normal return
		-1 = packing terms undefined

C-2

CHAPTER 5

ERROR (ER) LIBRARY

Error (ER) Library Summary

The error library is provided for processing errors from GEMPAK subroutines.

ER_WMSG writes error messages at the user's terminal. The message is written to the standard FORTRAN output unit 6.

The messages are stored in GEMPAK table files, which are sequential access files that can be created using any text editor. Each file may contain any number of leading comment records. These are records which begin with an exclamation point. Message records may contain up to 128 characters. They are free format and consist of the following fields separated by any number of spaces or tabs:

MESSAGE NUMBER
: The first field is the number that ER_WMSG uses to locate the message. It may be any non-zero integer value.

MESSAGE NAME
: The second field is a name that may be used for the message. This field is optional and is ignored by ER_WMSG.

MESSAGE
: The last field is the message to be printed. It must be preceded by an exclamation point which indicates the start of message. One !AS code may be included to indicate where a string is to be embedded. The code !\ may be used for a new line if a message is to appear on two lines. There is no provision for continuation lines within the file.

ER Library Calls

ER_WMSG (errgrp, numerr, errstr, / iret)

5.1 ER_WMSG - WRITE AN ERROR MESSAGE

This subroutine writes an error message to the user's terminal. The output message will contain the error group and error number in brackets followed by the message. If the error file or error number cannot be found, only the error group and number will be written.

The string, ERRSTR, will replace an !AS found in the message.

The messages are stored in error files. The message is read from the file GEMERR:'ERRGRP'.ERR.

ER_WMSG (ERRGRP, NUMERR, ERRSTR, IRET)

Input parameters:

ERRGRP	CHAR*	Error group
NUMERR	INTEGER	Error number
ERRSTR	CHAR*	String to be embedded

Output parameters:

IRET	INTEGER	Return code 3 = error number not found 2 = error file not found 0 = normal return

CHAPTER 6

FILE (FL) LIBRARY

FL_APND	Position file for append
FL_BKSP	Backspace sequential file
FL_CDEL	Close and delete file
FL_CLAL	Close all open files
FL_CLOS	Close file
FL_DCRE	Create direct access file
FL_DOPN	Open existing direct access file
FL_DSOP	Open shared direct access file
FL_FLUN	Free logical unit number
FL_GLUN	Allocate logical unit number
FL_GNAM	Get file name for logical unit
FL_IRET	Get GEMPAK error number
FL_INQR	Inquire whether a file exists
FL_PERR	Print I/O status error
FL_READ	Read direct access record
FL_REWD	Rewind sequential file
FL_RSHR	Read shared access record
FL_SOPN	Open sequential access file
FL_SUNK	Open unknown sequential file
FL_SWOP	Open sequential file for write
FL_TOPN	Open table file
FL_TREW	Rewind table file
FL_WRIT	Write direct access record

File (FL) Library Summary

The file library provides subroutines to access direct access files, sequential files and table files. The open and create subroutines return a logical unit number which can be used to access the files.

A table file is a sequential file which may have leading comment records. A comment record is any record where the first non-blank character is an exclamation point. The table open subroutine skips these leading comment records. Table files may be created using a text editor.

Direct access files may be created using FL_DCRE. The subroutines FL_DOPN and FL_DSOP open existing direct access files. FL_READ and FL_WRIT are provided to read and write data in direct access files.

The single subroutine FL_CLOS is provided for closing any file opened by an FL open subroutine.

Each of the FL subroutines returns a condition code, IRET, which is the GEMPAK file error number. This error number can be printed using ER_WMSG. If FORTRAN I/O services are called directly, the subroutine FL_IRET will translate the IOSTAT return into a GEMPAK file error number. The routine FL_PERR will translate the number and print an error message.

ERROR MESSAGES:

[FL 20]	REWIND error.
[FL 21]	Duplicate file specifications for file ....
[FL 22]	Input record too long.
[FL 23]	BACKSPACE error.
[FL 24]	End-of-file during read.
[FL 25]	Record number outside range.
[FL 26]	OPEN required for file ....
[FL 27]	Too many records in I/O statement.
[FL 28]	CLOSE error.
[FL 29]	File ... not found.
[FL 30]	Open failure for file ....
[FL 31]	Mixed file access modes for file ....
[FL 32]	Invalid logical unit number for file ....
[FL 33]	ENDFILE error.
[FL 34]	Unit already open.
[FL 36]	Attempt to access non-existent record.
[FL 37]	Inconsistent record length.
[FL 38]	File write error.

[FL 39]	File read error.
[FL 40]	Invalid recursive I/O operation.
[FL 41]	Insufficient virtual memory.
[FL 42]	Invalid device specification for file ....
[FL 43]	... is not a valid file specification.
[FL 44]	Inconsistent record type.
[FL 45]	Keyword value error in OPEN for file ....
[FL 46]	Inconsistent OPEN/CLOSE parameters for ....
[FL 47]	Write to READONLY file.
[FL 48]	Invalid argument to FORTRAN Run-Time Library.
[FL 51]	Inconsistent file organization for file ....
[FL 52]	Record locked.
[FL 53]	No current record.
[FL 54]	REWRITE error.
[FL 55]	DELETE error.
[FL 56]	UNLOCK error.
[FL 57]	FIND error.
[FL 62]	Syntax error in format for file ....
[FL 63]	Output conversion error.
[FL 64]	Input conversion error.
[FL 66]	Output statement overflows record.
[FL 67]	Input statement requires too much data.
[FL 68]	Variable format expression value error.
[FL -201]	The file ... is being used by another user.
[FL -202]	Invalid directory specification for file ....
[FL -203]	The file ... cannot be opened for write access.
[FL -204]	No more logical unit numbers available.
[FL -205]	No data records in table file ....
[FL +206]	No file open with logical unit number.
[FL -207]	Disk quota exceeded in creating file ....
[FL +208]	Logical unit number cannot be freed.
[FL -209]	Disk quota exceeded when extending file ....

FILE (FL) LIBRARY

FL Library Calls

```
FL_APND   ( lun, / iret )
FL_BKSP   ( lun, / iret )
FL_CDEL   ( lun, / iret )
FL_CLAL   ( / iret )
FL_CLOS   ( lun, / iret )
FL_DCRE   ( filnam, irecsz, / lun, iret )
FL_DOPN   ( filnam, irecsz, wrtflg, / lun, iret )
FL_DSOP   ( filnam, irecsz, / lun, iret )
FL_FLUN   ( lun, / iret )
FL_GLUN   ( / lun, iret )
FL_GNAM   ( lun, / filnam, iret )
FL_IRET   ( iostat, / iflerr, iret )
FL_INQR   ( filnam, / exist, iret )
FL_PERR   ( iostat, / msgnum, iret )
FL_READ   ( lun, irec, len, / iarray, iret )
FL_REWD   ( lun, / iret )
FL_RSHR   ( lun, irec, len, / iarray, iret )
FL_SOPN   ( filnam, / lun, iret )
FL_SUNK   ( filnam, / lun, iret )
FL_SWOP   ( filnam, / lun, iret )
FL_TOPN   ( filnam, / lun, iret )
FL_TREW   ( lun, / iret )
FL_WRIT   ( lun, irec, len, iarray, / iret )
```

6.1 FL_APND - POSITION FILE FOR APPEND

This subroutine positions a sequential file at the end-of-file mark so that records written after this call will be appended to the file.

FL_APND (LUN, IRET)

Input parameters:		
LUN	INTEGER	Logical unit number
Output parameters:		
IRET	INTEGER	Return code 0 = normal return

6.2 FL_BKSP - BACKSPACE SEQUENTIAL FILE

This subroutine backspaces a sequential file.

FL_BKSP (LUN, IRET)

Input parameters:

LUN	INTEGER	Logical unit number

Output parameters:

IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.3 FL_CDEL - CLOSE AND DELETE FILE

This subroutine closes and deletes a file that was opened by any FL subroutine and frees the assigned logical unit number. Note that this uses a non-standard FORTRAN option so that the file may not be deleted on UNIX systems.

FL_CDEL (LUN, IRET)

Input parameters:

LUN	INTEGER	Logical unit number

Output parameters:

IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.4 FL_CLAL - CLOSE ALL OPEN FILES

This subroutine closes all open files.

FL_CLAL (IRET)

Output parameters:

IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.5 FL_CLOS - CLOSE FILE

This subroutine closes a file that was opened by a FL subroutine and frees the assigned logical unit number.

FL_CLOS (LUN, IRET)

Input parameters:

LUN	INTEGER	Logical unit number

Output parameters:

IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.6 FL_DCRE - CREATE DIRECT ACCESS FILE

This subroutine creates a new direct access file and leaves the file open. It returns a logical unit number to be used to access the file.

FL_DCRE (FILNAM, IRECSZ, LUN, IRET)

Input parameters:

FILNAM	CHAR*	File name
IRECSZ	INTEGER	Record length in words

Output parameters:

LUN	INTEGER	Logical unit number
IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.7 FL_DOPN - OPEN EXISTING DIRECT ACCESS FILE

This subroutine opens an existing direct access file and returns a logical unit number to be used to access the file.

FL_DOPN (FILNAM, IRECSZ, WRTFLG, LUN, IRET)

Input parameters:

FILNAM	CHAR*	File name
IRECSZ	INTEGER	Record length in words
WRTFLG	LOGICAL	Write access flag

Output parameters:

LUN	INTEGER	Logical unit number
IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.8 FL_DSOP - OPEN SHARED DIRECT ACCESS FILE

This subroutine opens an existing direct access file for shared, write access. It returns a logical unit number to be used to access the file.

This subroutine is provided so that real-time data ingest programs can update a file while other programs have the file open for read access. FL_DOPN should be used to open files for write access by non-real-time programs.

FL_DSOP (FILNAM, IRECSZ, LUN, IRET)

Input parameters:

FILNAM	CHAR*	File name
IRECSZ	INTEGER	Record length in words

Output parameters:

LUN	INTEGER	Logical unit number
IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.9 FL_FLUN - FREE LOGICAL UNIT NUMBER

This subroutine frees a logical unit number that was allocated by FL_GLUN. A logical unit number should be freed when it is no longer needed.

FL_FLUN (LUN, IRET)

Input parameters:

LUN	INTEGER	Logical unit number

Output parameters:

IRET	INTEGER	Return code 0 = normal return <>0 = lun could not be freed

6.10 FL_GLUN - ALLOCATE LOGICAL UNIT NUMBER

This subroutine gets a logical unit number that can be used for file access. It is used to eliminate conflicts in assigning logical unit numbers.

FL_GLUN (LUN, IRET)

Output parameters:

LUN	INTEGER	Logical unit number
IRET	INTEGER	Return code 0 = normal return -204 = no more luns

6.11 FL_GNAM - GET FILE NAME FOR LOGICAL UNIT

This subroutine returns the file name associated with a logical unit number.

FL_GNAM (LUN, FILNAM, IRET)

Input parameters:

LUN	INTEGER	Logical unit number

Output parameters:

FILNAM	CHAR*	File name
IRET	INTEGER	Return code 0 = normal return 206 = unit not open <>0 = GEMPAK file error

6.12 FL_IRET - GET GEMPAK ERROR NUMBER

This subroutine takes the IOSTAT value returned from a FORTRAN I/O statement and determines the GEMPAK message number for the error. This value can be used to write a GEMPAK FL error message. This subroutine must be called immediately after the I/O operation.

FL_IRET (IOSTAT, IFLERR, IRET)

Input parameters:

IOSTAT	INTEGER	Status from I/O operation

Output parameters:

IFLERR	INTEGER	GEMPAK file error
IRET	INTEGER	Return code 0 = normal return

6.13 FL_INQR - INQUIRE WHETHER A FILE EXISTS

This subroutine determines whether a file exists.

FL_INQR (FILNAM, EXIST, IRET)

Input parameters:

FILNAM	CHAR*	File name

Output parameters:

EXIST	LOGICAL	File exists flag
IRET	INTEGER	Return code 0 = normal return

6.14 FL_PERR - PRINT I/O STATUS ERROR

This subroutine translates errors returned as IOSTAT values in Fortran I/O statement into GEMPAK error numbers and prints the error message. Errors returned from FL subroutines are already translated and may be printed using ER_WMSG.

FL_PERR (IOSTAT, MSGNUM, IRET)

Input parameters:

IOSTAT	INTEGER	Status returned I/O operation

Output parameters:

MSGNUM	INTEGER	GEMPAK file error
IRET	INTEGER	Return code 0 = normal return

6.15 FL_READ - READ DIRECT ACCESS RECORD

This subroutine reads a record from a direct access file. On VMS systems, if the record is locked by another user, 30 tries to open the file will be attempted at 1-second intervals.

FL_READ (LUN, IREC, LEN, IARRAY, IRET)

Input parameters:

LUN	INTEGER	Logical unit number
IREC	INTEGER	Record number
LEN	INTEGER	Record length in words

Output parameters:

IARRAY (LEN)	INTEGER	Data record
IRET	INTEGER	Return code 0 = normal return 52 = locked record <>0 = GEMPAK file error

6.16 FL_REWD - REWIND SEQUENTIAL FILE

This subroutine rewinds a sequential file.

FL_REWD (LUN, IRET)

Input parameters:

LUN	INTEGER	Logical unit number

Output parameters:

IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.17 FL_RSHR - READ SHARED ACCESS RECORD

This subroutine reads a record from a direct access file. On a VMS system, if the record is locked by another user, 30 tries to open the file will be attempted at 1-second intervals. This subroutine is meant to be called when a file is opened for shared, write access. As each record is read, it is written back to the file in order to prevent records from being locked on VMS systems. This subroutine should not be necessary on UNIX systems.

FL_RSHR (LUN, IREC, LEN, IARRAY, IRET)

Input parameters:

LUN	INTEGER	Logical unit number
IREC	INTEGER	Record number
LEN	INTEGER	Record length in words

Output parameters:

IARRAY (LEN)	INTEGER	Data record
IRET	INTEGER	Return code 0 = normal return 52 = locked record <>0 = GEMPAK file error

6.18 FL_SOPN - OPEN SEQUENTIAL ACCESS FILE

This subroutine opens an existing sequential file and returns a logical unit number to be used to access the file. The file is opened as a READONLY file.

FL_SOPN (FILNAM, LUN, IRET)

Input parameters:

FILNAM	CHAR*	File name

Output parameters:

LUN	INTEGER	Logical unit number
IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.19 FL_SUNK - OPEN UNKNOWN SEQUENTIAL FILE

This subroutine opens an sequential file and returns a logical unit number to be used to access the file. The file is opened as a new file, if possible. If not, it is opened with status of unknown. Thus, a new version will be created on VMS systems and the existing file will be rewritten on UNIX systems.

FL_SUNK (FILNAM, LUN, IRET)

Input parameters:

FILNAM	CHAR*	File name

Output parameters:

LUN	INTEGER	Logical unit number
IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.20 FL_SWOP - OPEN SEQUENTIAL FILE FOR WRITE

This subroutine opens or creates a sequential file and returns a logical unit number to be used to access the file. The file is opened for write access.

FL_SWOP (FILNAM, LUN, IRET)

Input parameters:

FILNAM	CHAR*	File name

Output parameters:

LUN	INTEGER	Logical unit number
IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.21 FL_TOPN - OPEN TABLE FILE

This subroutine opens an existing table file. A table file is a sequential file that may have comment records at the beginning of the file. If the first non-blank character in the first 80 characters is an exclamation point, the record is a comment record. Leading comment records are skipped and the file is positioned for reading at the first valid data record. The file is opened for READONLY access.

FL_TOPN (FILNAM, LUN, IRET)

Input parameters:

FILNAM	CHAR*	File name

Output parameters:

LUN	INTEGER	Logical unit number
IRET	INTEGER	Return code 0 = normal return -205 = no data records found <>0 = GEMPAK file error

6.22 FL_TREW - REWIND TABLE FILE

This subroutine rewinds a table file that was opened by FL_TOPN. The file is positioned to read the first data record in the file.

FL_TREW (LUN, IRET)

Input parameters:

LUN	INTEGER	Logical unit number

Output parameters:

IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

6.23 FL_WRIT - WRITE DIRECT ACCESS RECORD

This subroutine writes a record to a direct access file.

FL_WRIT (LUN, IREC, LEN, IARRAY, IRET)

Input parameters:

LUN	INTEGER	Logical unit number
IREC	INTEGER	Record number
LEN	INTEGER	Record length in words
IARRAY (LEN)	INTEGER	Data record

Output parameters:

IRET	INTEGER	Return code 0 = normal return <>0 = GEMPAK file error

CHAPTER 7

GRID (GD) LIBRARY

GD_CLOS	Close grid file
GD_CREF	Create grid file
GD_DGRD	Delete grid from file
GD_GANL	Get analysis block
GD_GGRD	Read grid by number
GD_GIDN	Read grid identifier
GD_GLEV	Get grid levels
GD_GNAV	Get navigation block
GD_GNUM	Get grid number
GD_GTIM	Get grid times
GD_NGRD	Return number of grids
GD_OPNF	Open grid file
GD_OPNR	Open realtime grid file
GD_RDAT	Read grid from file
GD_SWRT	Set write flag in grid file
GD_WDAT	Write grid to file
GD_WPGD	Write packed grid
GD_WPPG	Write pre-packed grid

GRID (GD) LIBRARY

Grid (GD) Library Summary

The grid library subroutines allow the programmer to access GEMPAK grid files. Subroutines are available to create new files and to read and write information in existing files.

A grid file is a collection of grids; each grid is a two-dimensional array of numbers. In general, each grid represents a quasi-horizontal slice through the atmosphere. Each grid in the file has a grid identifier containing the time, vertical level, vertical coordinate system and parameter name.

GRID IDENTIFIER:

TIME   CHARACTER*20 (2)

Time is formatted as the GEMPAK standard grid time,

YYMMDD/HHMMthhhmm

where:

YYMMDD	is the year, month, day	
HHMM	is the hour, minute	
t	is the type (F=forecast, A=analysis, G=guess)	
hhhmm	is the forecast hour, minute.	

Two time fields may be included in the grid identifier. These may be used, for example, for the difference of two times. If only a single time is needed, TIME (2) = ' '. If t is blank, an analysis grid is assumed. If hhhmm is blank, 00000 is assumed. If hhhmm has one or two digits, they represent hours. With three or more digits, zeros will be added at the beginning of the field.

VERTICAL LEVEL   INTEGER (2)

The vertical level part of the grid identifier is stored as two integers. If only a single level is needed, the second value is set to -1.

VERTICAL COORDINATE   INTEGER

The vertical coordinate is stored as an integer with the following values:

0 = NONE
1 = PRESSURE

2 = THETA
3 = HEIGHT

PARAMETER NAME   CHARACTER*12

For the basic meteorological parameters, the 4-character GEMPAK name is used.

A grid may also be identified by a grid number, which is its current position in the grid file. Use of the grid number may be convenient when selecting grids from a list. However, since grids are sorted before they are numbered, the number corresponding to a grid may change when grids are added to or deleted from the file.

GRID NAVIGATION BLOCK:

All the grids in a file must be co-located--that is, the information locating the grid on the earth is defined once for the entire file. The grid points must be evenly spaced in some coordinate system. This location information is stored in a grid navigation block. The subroutine GR_MNAV will pack the navigation information into a navigation block. The navigation block should be declared 256 words long.

Following is a list of the contents of a grid navigation block. Note that an evenly spaced latitude/longitude grid has projection type "CED". The numbers are all real numbers.

WORD	CONTENTS	
1	Grid definition type 1 = simple map projection 2 = full map projection 3 = graph	
2	Projection	(3-char name packed in real word)
3	Left grid number	(always 1)
4	Bottom grid number	(always 1)
5	Right grid number	(KX)
6	Top grid number	(KY)
7	Lower left latitude	
8	Lower left longitude	
9	Upper right latitude	
10	Upper right longitude	
11	Projection angle 1	
12	Projection angle 2	
13	Projection angle 3	
14-256	Spares	

GRID ANALYSIS BLOCK:

In addition to the grid navigation block, a single grid analysis block may be saved with each file. This block contains information used in performing an objective analysis. The subroutine GR_MBAN packs information into a Barnes analysis block. The analysis block should be declared to be 128 words long.

The grid analysis block for a Barnes analysis contains the following information. The numbers are all real numbers.

WORD	CONTENTS FOR BARNES ON CED GRID
1	Analysis type = 1.0
2	Deltan
3	Deltax
4	Deltay
5	Not used
6-9	Grid area bounds
10-13	Extend area bounds
14-17	Data area bounds
18-128	Spares

WORD	CONTENTS FOR GENERAL BARNES
1	Analysis type = 2.0
2	Deltan
3-6	Grid extension (grid units)
7-10	Grid area
11-14	Extend area bounds
15-18	Data area bounds
19-128	Spares

GRID HEADER BLOCK:

A grid header block may also be saved with each grid. This header contains information about the particular grid. The GEMPAK grid header contains two (integer) words to store the offset in half-grid units of the current grid from the base grid defined by the navigation block. No GEMPAK programs currently use these words.

WORD	CONTENTS
1	X offset in half-grid units
2	Y offset in half-grid units

ERROR MESSAGES:

GRID (GD) LIBRARY

[GD -1]	File ... cannot not be created.
[GD -2]	File ... cannot not be opened.
[GD -3]	File cannot be closed.
[GD -4]	File not open.
[GD -5]	No write access to file.
[GD -6]	File read/write error.
[GD -7]	File ... is not a GEMPAK grid file.
[GD -8]	Grid navigation block cannot be read.
[GD -9]	Invalid grid size.
[GD -10]	Grid already exists.
[GD -11]	Grid file is full.
[GD -12]	Grid does not exist.
[GD -13]	Grid header length is too long.

GD Library Calls

GD_CLOS (igdfln, / iret)

GD_CREF (filnam, navsz, rnvblk, ianlsz, anlblk, ihdrsz, maxgrd, / igdfln, iret)

GD_DGRD (igdfln, gdattm, level, ivcord, parm, / iret)

GD_GANL (igdfln, / anlblk, ianlsz, iret)

GD_GGRD (igdfln, ignum, / gdattm, level, ivcord, parm, grid, igx, igy, ighdr, iret)

GD_GIDN (igdfln, ignum, / gdattm, level, ivcord, parm, iret)

GD_GLEV (igdfln, gdattm, ivcord, maxlev, / levarr, nlev, iret)

GD_GNAV (igdfln, / rnvblk, navsz, iret)

GD_GNUM (igdfln, gdattm, level, ivcord, parm, / ignum, iret)

GD_GTIM (igdfln, maxtim, / timarr, ntimes, iret)

GD_NGRD (igdfln, / numgrd, firstm, lasttm, iret)

GD_OPNF (filnam, wrtflg, / igdfln, navsz, rnvblk, ianlsz, anlblk, ihdrsz, maxgrd, iret)

GD_OPNR (filnam, / igdfln, navsz, rnvblk, ianlsz, anlblk, ihdrsz, maxgrd, iret)

GD_RDAT (igdfln, gdattm, level, ivcord, parm, / grid, igx, igy, ighdr, iret)

GD_SWRT (igdfln, wrtflg, / iret)

GD_WDAT (igdfln, grid, igx, igy, ighdr, gdattm, level, ivcord, parm, rewrit, / iret)

GD_WPGD (igdfln, grid, igx, igy, ighdr, gdattm, level, ivcord, parm, rewrit, ipktyp, nbits, / iret)

GD_WPPG (igdfln, igrid, lengrd, igx, igy, ighdr, gdattm, level, ivcord, parm, rewrit, ipktyp, nbits, misflg, ref, scale, difmin, / iret)

7.1 GD_CLOS - CLOSE GRID FILE

This subroutine closes a grid file.

GD_CLOS (IGDFLN, IRET)

Input parameters:

IGDFLN	INTEGER	File number

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file can't be closed -4 = file not open

7.2 GD_CREF - CREATE GRID FILE

This subroutine creates a new GEMPAK5 grid file. If MAXGRD is zero or negative, it will default to 400. IHDRSZ is the length of the grid header which will be stored with every grid. This header is intended to save offsets from a base grid, but is not currently used. IHDRSZ should usually be set to 2.

```
GD_CREF  ( FILNAM, NAVSZ, RNVBLK, IANLSZ, ANLBLK, IHDRSZ,
           MAXGRD, IGDFLN, IRET )
```

Input parameters:

FILNAM	CHAR*	File name
NAVSZ	INTEGER	Navigation block length (256)
RNVBLK (NAVSZ)	REAL	Navigation block
IANLSZ	INTEGER	Analysis block length (128)
ANLBLK (IANLSZ)	REAL	Analysis block
IHDRSZ	INTEGER	Grid header length
MAXGRD	INTEGER	Max number of grids in file

Output parameters:

IGDFLN	INTEGER	Grid file number
IRET	INTEGER	Return code 0 = normal return -1 = file cannot be created -13 = grid header too long

7.3 GD_DGRD - DELETE GRID FROM FILE

This subroutine deletes a grid from a grid file.

GD_DGRD (IGDFLN, GDATTM, LEVEL, IVCORD, PARM, IRET)

Input parameters:

IGDFLN	INTEGER	Grid file number
GDATTM (2)	CHAR*20	GEMPAK time
LEVEL (2)	INTEGER	Vertical level
IVCORD	INTEGER	Vertical coordinate 0 = none 1 = PRES 2 = THTA 3 = HGHT
PARM	CHAR*12	Parameter name

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -5 = no write access to file -6 = read/write error -12 = grid does not exist

7.4 GD_GANL - GET ANALYSIS BLOCK

This subroutine returns the analysis block.

GD_GANL (IGDFLN, ANLBLK, IANLSZ, IRET)

Input parameters:

IGDFLN	INTEGER	Grid file number

Output parameters:

ANLBLK (IANLSZ)	REAL	Analysis block
IANLSZ	INTEGER	Length of anl block
IRET	INTEGER	Return code 0 = normal return

7.5 GD_GGRD - READ GRID BY NUMBER

This subroutine reads the requested grid from a grid file given the grid number.

GD_GGRD (IGDFLN, IGNUM, GDATTM, LEVEL, IVCORD, PARM, GRID, IGX,
IGY, IGHDR, IRET)

Input parameters:

IGDFLN	INTEGER	Grid file number
IGNUM	INTEGER	Grid number

Output parameters:

GDATTM (2)	CHAR*20	GEMPAK times
LEVEL (2)	INTEGER	Vertical levels
IVCORD	INTEGER	Vertical coordinate
PARM	CHAR*12	Parameter name
GRID (IGX,IGY)	REAL	Grid data
IGX	INTEGER	Number of horizontal points
IGY	INTEGER	Number of vertical points
IGHDR (IHDRSZ)	INTEGER	Grid header
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = read error -12 = grid does not exist

7.6 GD_GIDN - READ GRID IDENTIFIER

This subroutine returns a grid identifier given the grid number.

GD_GIDN (IGDFLN, IGNUM, GDATTM, LEVEL, IVCORD, PARM, IRET)

Input parameters:

IGDFLN	INTEGER	Grid file number
IGNUM	INTEGER	Grid number

Output parameters:

GDATTM (2)	CHAR*20	GEMPAK times
LEVEL (2)	INTEGER	Vertical levels
IVCORD	INTEGER	Vertical coordinate
PARM	CHAR*12	Parameter name
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = read/write error -12 = invalid grid number

7.7 GD_GLEV - GET GRID LEVELS

This subroutine returns all the levels present in a grid file for a given date and vertical coordinate. The levels returned are not sorted.

GD_GLEV (IGDFLN, GDATTM, IVCORD, MAXLEV, LEVARR, NLEV, IRET)

Input parameters:

IGDFLN	INTEGER	Grid file number
GDATTM (2)	CHAR*20	GEMPAK times
IVCORD	INTEGER	Vertical coordinate
MAXLEV	INTEGER	Maximum number of levels

Output parameters:

LEVARR (2,NLEV)	INTEGER	Levels found
NLEV	INTEGER	Number of levels found
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = read/write error

7.8 GD_GNAV - GET NAVIGATION BLOCK

This subroutine returns the navigation block.

GD_GNAV (IGDFLN, RNVBLK, NAVSZ, IRET)

Input parameters:

IGDFLN	INTEGER	Grid file number

Output parameters:

RNVBLK (NAVSZ)	REAL	Navigation block
NAVSZ	INTEGER	Length of nav block
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = read/write error

7.9 GD_GNUM - GET GRID NUMBER

This subroutine gets the grid number for the requested grid.

GD_GNUM (IGDFLN, GDATTM, LEVEL, IVCORD, PARM, IGNUM, IRET)

Input parameters:

IGDFLN	INTEGER	File number
GDATTM (2)	CHAR*20	GEMPAK times
LEVEL (2)	INTEGER	Vertical levels
IVCORD	INTEGER	Vertical coordinate 0 = NONE 1 = PRES 2 = THTA 3 = HGHT
PARM	CHAR*12	Parameter name

Output parameters:

IGNUM	INTEGER	Grid number
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = read/write error -12 = grid does not exist

7.10 GD_GTIM - GET GRID TIMES

This subroutine returns all the times present in a grid file. Only the first times are returned. They are sorted from earliest to latest.

GD_GTIM (IGDFLN, MAXTIM, TIMARR, NTIMES, IRET)

Input parameters:

IGDFLN	INTEGER	Grid file number
MAXTIM	INTEGER	Maximum number of times

Output parameters:

TIMARR (NTIMES)	CHAR*	GEMPAK times
NTIMES	INTEGER	Number of times
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = read/write error

7.11 GD_NGRD - RETURN NUMBER OF GRIDS

This subroutine returns the number of grids in a grid file along with the first and last time.

GD_NGRD (IGDFLN, NUMGRD, FIRSTM, LASTTM, IRET)

Input parameters:

IGDFLN	INTEGER	Grid file number

Output parameters:

NUMGRD	INTEGER	Number of grids
FIRSTM	CHAR*20	Earliest time1 in file
LASTTM	CHAR*20	Latest time1 in file
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = read error

7.12 GD_OPNF - OPEN GRID FILE

This subroutine opens an existing GEMPAK grid file. If the file requires shared, write access, the subroutine GD_OPNR should be used.

```
GD_OPNF ( FILNAM, WRTFLG, IGDFLN, NAVSZ, RNVBLK, IANLSZ, ANLBLK,
          IHDRSZ, MAXGRD, IRET )
```

Input parameters:

FILNAM	CHAR*	File name
WRTFLG	LOGICAL	Flag for write access

Output parameters:

IGDFLN	INTEGER	File number
NAVSZ	INTEGER	Navigation block length
RNVBLK (NAVSZ)	REAL	Navigation block
IANLSZ	INTEGER	Analysis block length
ANLBLK (IANLSZ)	REAL	Analysis block
IHDRSZ	INTEGER	Grid header length
MAXGRD	INTEGER	Maximum number of grids
IRET	INTEGER	Return code 0 = normal return -2 = file cannot be opened -7 = not a GEMPAK5 grid file -8 = nav cannot be read -13 = grid header too long -14 = file name is blank

7.13 GD_OPNR - OPEN REALTIME GRID FILE

This subroutine opens an existing GEMPAK grid file for realtime data access. The file is opened with shared, write access.

GD_OPNR (FILNAM, IGDFLN, NAVSZ, RNVBLK, IANLSZ, ANLBLK, IHDRSZ, MAXGRD, IRET)

Input parameters:

FILNAM	CHAR*	File name

Output parameters:

IGDFLN	INTEGER	File number
NAVSZ	INTEGER	Navigation block length
RNVBLK (NAVSZ)	REAL	Navigation block
IANLSZ	INTEGER	Analysis block length
ANLBLK (IANLSZ)	REAL	Analysis block
IHDRSZ	INTEGER	Grid header length
MAXGRD	INTEGER	Maximum number of grids
IRET	INTEGER	Return code 0 = normal return -2 = file cannot be opened -7 = not a GEMPAK5 grid file -8 = nav cannot be read -13 = grid header too long -14 = file name is blank

7.14 GD_RDAT - READ GRID FROM FILE

This subroutine reads the requested grid from a grid file.

```
GD_RDAT  ( IGDFLN, GDATTM, LEVEL, IVCORD, PARM, GRID, IGX, IGY,
           IGHDR,  IRET )
```

Input parameters:

IGDFLN	INTEGER	Grid file number
GDATTM (2)	CHAR*20	GEMPAK times
LEVEL (2)	INTEGER	Vertical levels
IVCORD	INTEGER	Vertical coordinate 0 = NONE 1 = PRES 2 = THTA 3 = HGHT
PARM	CHAR*12	Parameter name

Output parameters:

GRID (IGX,IGY)	REAL	Grid data
IGX	INTEGER	Number of horizontal points
IGY	INTEGER	Number of vertical points
IGHDR (IHDRSZ)	INTEGER	Grid header
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = read/write error -12 = grid does not exist

7.15 GD_SWRT - SET WRITE FLAG IN GRID FILE

This subroutine sets the internal write flag for a grid file. If the file is being changed from READ ONLY to WRITE access, DM_CHNG will close it and reopen it for WRITE access.

GD_SWRT (IGDFLN, WRTFLG, IRET)

Input parameters:

IGDFLN	INTEGER	Grid file number
WRTFLG	LOGICAL	Write flag (T = write)

Output parameters:

IRET	INTEGER	Return code 0 = normal return

7.16 GD_WDAT - WRITE GRID TO FILE

This subroutine writes a grid into a grid file.

```
GD_WDAT  ( IGDFLN, GRID, IGX, IGY, IGHDR, GDATTM, LEVEL, IVCORD,
           PARM, REWRIT, IRET )
```

Input parameters:

IGDFLN	INTEGER	Grid file number
GRID (IGX,IGY)	REAL	Grid data
IGX	INTEGER	Number of horizontal points
IGY	INTEGER	Number of vertical points
IGHDR (IHDRSZ)	INTEGER	Grid header
GDATTM (2)	CHAR*20	GEMPAK times
LEVEL (2)	INTEGER	Vertical levels
IVCORD	INTEGER	Vertical coordinate 0 = NONE 1 = PRES 2 = THTA 3 = HGHT
PARM	CHAR*12	Parameter name
REWRIT	LOGICAL	Flag to replace existing grid

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -5 = no write access -6 = read/ write error -9 = invalid grid size -10 = grid already exists -11 = grid file is full

7.17 GD_WPGD - WRITE PACKED GRID

This subroutine packs an input grid of real values and writes it to a grid file. IPKTYP should be one of the following parameter names from GEMPRM.PRM:

MDGNON	No grid packing
MDGGRB	Pack in GEMPAK GRIB format given nbits
MDGDEC	Pack in GEMPAK GRIB format given precision
MDGDIF	Pack in GEMPAK DIF format given nbits

If the packing type is MDGNON, the real data will be stored as if GD_WDAT were called. If MDGGRB or MDGDIF is specified, the number of bits given in NBITS will be used to store the data. For packing type MDGDEC, NBITS is the precision. The grid data is multiplied by 10 ** NBITS and rounded to the nearest integer. The actual number of bits used to store the data is the minimum number required to store the resulting integers.

```
GD_WPGD  ( IGDFLN, GRID, IGX, IGY, IGHDR, GDATTM, LEVEL, IVCORD,
           PARM, REWRIT, IRET )
```

Input parameters:

IGDFLN	INTEGER	Grid file number
GRID (IGX,IGY)	REAL	Grid data
IGX	INTEGER	Number of horizontal points
IGY	INTEGER	Number of vertical points
IGHDR (IHDRSZ)	INTEGER	Grid header
GDATTM (2)	CHAR*20	GEMPAK times
LEVEL (2)	INTEGER	Vertical levels
IVCORD	INTEGER	Vertical coordinate 0 = NONE 1 = PRES 2 = THTA 3 = HGHT
PARM	CHAR*12	Parameter name
REWRIT	LOGICAL	Flag to replace existing grid
IPKTYP	INTEGER	Packing type
NBITS	INTEGER	Number of bits / precision

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -5 = no write access -6 = read/ write error -9 = invalid grid size -10 = grid already exists -11 = grid file is full

7.18 GD_WPPG - WRITE PRE-PACKED GRID

This subroutine writes a grid that is already packed to a grid file. IPKTYP should be one of the following parameter names:

```
        MDGGRB         Packed in GEMPAK GRIB format
                         REF    = minimum value
                         SCALE  = 2 ** N
        MDGNMC         Packed in NMC format
                         REF    = average value
                         SCALE  = 1 / 2 ** N
        MDGDIF         Packed in GEMPAK DIF format
                         REF    = first non-missing point in grid
                         SCALE  = scaling term for differences
                         DIFMIN = minimum value of difference field
```

```
GD_WPPG  ( IGDFLN, IGRID, LENGRD, IGX, IGY, IGHDR, GDATTM, LEVEL,
           IVCORD, PARM, REWRIT, IPKTYP, NBITS, MISFLG, REF,
           SCALE, DIFMIN, IRET )
```

Input parameters:

IGDFLN	INTEGER	Grid file number
GRID (IGX,IGY)	REAL	Grid data
IGX	INTEGER	Number of horizontal points
IGY	INTEGER	Number of vertical points
IGHDR (IHDRSZ)	INTEGER	Grid header
GDATTM (2)	CHAR*20	GEMPAK times
LEVEL (2)	INTEGER	Vertical levels
IVCORD	INTEGER	Vertical coordinate 0 = NONE 1 = PRES 2 = THTA 3 = HGHT
PARM	CHAR*12	Parameter name
REWRIT	LOGICAL	Flag to replace existing grid
IPKTYP	INTEGER	Packing type
NBITS	INTEGER	Number of bits
MISFLG	LOGICAL	Missing data flag
REF	REAL	Reference value
SCALE	REAL	Scaling factor
DIFMIN	REAL	DIF reference value

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -5 = no write access -6 = read/ write error -9 = invalid grid size -10 = grid already exists -11 = grid file is full

CHAPTER 8

GRAPHICS (GG) LIBRARY

GG_BOX	Draw box
GG_INIT	Initialize GEMPLT
GG_LTLN	Draw lat/lon grid
GG_MAP	Draw map
GG_PANL	Define view region
GG_PROJ	Process PROJ input
GG_SAOI	Define AOIPS satellite nav
GG_SDEV	Set graphics device
GG_SGRF	Define graph coordinate system
GG_SKEW	Set up skew T plot
GG_SMAP	Set map projection
GG_SNPG	Define NPGS satellite nav
GG_WSTR	Write title

Graphics (GG) Library Summary

The graphics library is used to simplify and standardize GEMPLT library calls. Routines to initialize graphics, to set the graphics device and projection, and to draw maps and titles are included.

GG_SMAP is used to define both the projection type and graphics area. It can be used for map, graph and satellite overlay projections. The current valid projections are listed in the documentation for GG_SMAP. Details for defining map projections can be found in the GEMPLT documentation for GSMMAP and GSMPRJ.

ERROR MESSAGES:

[GG -1]	Invalid mode set.
[GG -2]	Area ... is an invalid graphics area.
[GG -3]	Error initializing GEMPLT.
[GG -4]	Error in graph mode setup.
[GG -5]	Projection ... is invalid.
[GG -6]	Device ... is invalid.
[GG -7]	No map drawn.
[GG -8]	Margins requested with NM.
[GG -9]	Invalid region specified.
[GG -10]	Panel not recognized.
[GG -11]	Error in setting view.

GRAPHICS (GG) LIBRARY

GG Library Calls

```
GG_BOX    ( region, icolor, ilntyp, ilnwid, / iret )
GG_INIT   ( mode, / iret )
GG_LTLN   ( latlon, / iret )
GG_MAP    ( map, / iret )
GG_PANL   ( panel, / iret )
GG_PROJ   ( proj, / cprj, angle, zmarg, angflg, iret )
GG_SAOI   ( garea, / iret )
GG_SDEV   ( device, / iret )
GG_SGRF   ( proj, garea, / iret )
GG_SKEW   ( xaxis, yaxis, parm, / ratio, xstrt, ystrt, xstop, ystop,
            xlbl, nxlbl, / iret )
GG_SMAP   ( proj, garea, / iret )
GG_SNPG   ( garea, / iret )
GG_WSTR   ( string, line, / iret )
```

8.1 GG_BOX - DRAW BOX

This subroutine draws a box around the specified area. If the color is zero, no box is drawn. If the line type is zero, the default line type is used. If the width is 0, a width of 1 is set.

GG_BOX (REGION, ICOLOR, ILNTYP, ILNWID, IRET)

Input parameters:

REGION	CHAR*	Coordinate region 'D' = device 'N' = normalized 'V' = view 'P' = plot
ICOLOR	INTEGER	Color number
ILNTYP	INTEGER	Line type
ILNWID	INTEGER	Line width

Output parameters:

IRET	INTEGER	Return code 0 = normal return -9 = invalid region

8.2 GG_INIT - INITIALIZE GEMPLT

This subroutine initializes the GEMPLT plotting package. The current map file is set to the global map file name found in $MAPFIL. Thus, it is necessary to call IP_INIT before calling this subroutine. If IP_INIT has not been called, the map file will not be defined.

In the past, this subroutine set default margins for map and graph mode. Currently, margins will not be set or changed in GG_INIT. Margins can be specified by the user in the input for PROJ. The margin definition will be extracted by GG_PROJ and set in GG_SMAP.

GG_INIT (MODE, IRET)

Input parameters:

MODE	INTEGER	Plot mode 0 = no change 1 = map 2 = graph

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = error starting GEMPLT

8.3 GG_LTLN - DRAW LAT/LON GRID

This subroutine draws latitude/longitude lines on the graphics device. The LATLON string should contain the line color, line type, line width, label frequency and latlon increment information separated by slashes. The latter consists of the latitude and longitude increments separated by semicolons. If LATLON is blank, no lines will be drawn

GG_LTLN (LATLON, IRET)

Input parameters:

LATLON	CHAR*	Line col/typ/wdth/lblfr/inc

Output parameters:

IRET	INTEGER	Return code 0 = normal return -13 = lines not drawn

8.4 GG_MAP - DRAW MAP

This subroutine draws a map on the graphics device. The MAP string should contain the map color, line type and line width separated by slashes (/). If the line type or width is zero or undefined, the current value is used. If MAP is a blank, a default color of 1 will be used.

GG_MAP (MAP, IRET)

Input parameters:

MAP	CHAR*	Map color/line type/width

Output parameters:

IRET	INTEGER	Return code 0 = normal return -7 = map not drawn

8.5 GG_PANL - DEFINE VIEW REGION

This subroutine sets the view region for the panel specified. If requested, a box will be drawn around the region.

The input for PANEL specifies the panel location, panel outline color, line type and width separated with slashes. The panel location determines the location of the view region on the graphics device. It may be specified using a number or abbreviation as follows:

NUMBER	ABBREVIATION	DESCRIPTION
0	ALL	Entire device
1	UL	Upper left quadrant
2	UR	Upper right quadrant
3	LL	Lower left quadrant
4	LR	Lower right quadrant
5	L	Left half
6	R	Right half
7	T	Top half
8	B	Bottom half

Horizontal or vertical panels which divide the screen into thirds or fourths may be created using the syntax Tij where T is either V for vertical or H for horizontal, i is 3 for thirds or 4 for fourths, and j is the actual panel counting from the top or left.

The view region may also be specified as four numbers separated with semicolons, giving the lower left and upper right corners in fractions of the graphics display area.

GG_PANL (PANEL, IRET)

Input parameters:

PANEL	CHAR*	Input for PANEL

Output parameters:

IRET	INTEGER	Return code 0 = normal return -9 = invalid region -10 = panel not recognized -11 = error in setting view

8.6 GG_PROJ - PROCESS PROJ INPUT

This subroutine decodes the user input for the parameter PROJ. The input may contain parts separated with slashes. The first part must be the projection name. Other parts may include:

NM	-	margins will be set to 0
3 numbers	-	angles for a full map projection
4 numbers	-	margins

If angles are input, ANGFLG will be set to indicate that a full map projection was specified. If margins are not input and NM is also not included in the string, default margins will be set. The default for map projections is (0,3,0,0) and for graphs is (6,4,4,1). A complete description of projections and margins can be found in the GEMPLT Programmer's Guide.

GG_PROJ (PROJ, CPRJ, ANGLE, ZMARG, ANGFLG, IRET)

Input parameters:

PROJ	CHAR*	Input projection string

Output parameters:

CPRJ	CHAR*	Projection name
ANGLE (3)	REAL	Projection angles
ZMARG (4)	REAL	Margins
ANGFLG	LOGICAL	Angle flag
IRET	INTEGER	Return code 0 = normal return

8.7 GG_SAOI - DEFINE AOIPS SATELLITE NAV

This subroutine sets the satellite navigation for an AOIPS image. In this case, GAREA is the name of an AOIPS image file.

GG_SAOI (GAREA, IRET)

Input parameters:

GAREA	CHAR*	Image name

Output parameters:

IRET	INTEGER	Return code 0 = normal return -5 = invalid projection

8.8 GG_SDEV - SET GRAPHICS DEVICE

This subroutine sets the graphics device in GEMPLT. If an error is returned from GEMPLT, an error message is written.

GG_SDEV (DEVICE, IRET)

Input parameters:

DEVICE	CHAR*	Device name

Output parameters:

IRET	INTEGER	Return code 0 = normal return -6 = invalid device specified

8.9 GG_SGRF - DEFINE GRAPH COORDINATE SYSTEM

This subroutine defines an output graph coordinate system. PROJ must be:

POL	polar coordinates
LIN	linear x and y
LOG	linear x, logarithmic y
KAP	linear x, y ** KAPPA (KAPPA = 2 / 7)

The graphics area GAREA is specified by five numbers corresponding to the lower left x, lower left y, upper right x, upper right y and the height-to-width ratio of the plotting area. If the plot ratio is unspecified or 0, the entire area inside the margins will be used.

```
GG_SGRF ( PROJ, GAREA, IRET )
```

Input parameters:

PROJ	CHAR*	Projection type
GAREA	CHAR*	Graphics area

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = error specifying graph -5 = invalid projection

8.10 GG_SKEW - SET UP SKEW T PLOT

This subroutine sets the graphics for a skew T plot. The aspect ratio is computed. A section of a standard skew T is determined and GSGRAF is called. IN_AXIS should be called first for both XAXIS and YAXIS to establish user input or default bounds.

```
GG_SKEW  ( XAXIS, YAXIS, PARM, RATIO, XSTRT, YSTRT, XSTOP,
           YSTOP, XLBL, NXLBL, IRET )
```

Input parameters:

XAXIS	CHAR*	Input for X axis
YAXIS	CHAR*	Input for Y axis
PARM	CHAR*	Parameter list or function

Input and output parameters:

RATIO	REAL	Aspect ratio
XSTRT	REAL	Minimum on T axis
YSTRT	REAL	Maximum on P axis
XSTOP	REAL	Maximum on T axis
YSTOP	REAL	Minimum on P axis
XLBL (NXLBL)	REAL	Label values for T axis
NXLBL	INTEGER	Number of label values

Output parameters:

IRET	INTEGER	Return code 0 = normal return -12 = no temperature parm

8.11 GG_SMAP - SET MAP PROJECTION

This subroutine defines the map or graph projection and graphics area in GEMPLT. If a GEMPLT error is encountered, an error message is written. If PROJ = DEF, the current map projection will be retained. No validity check will be made.

The following simple map projections may be specified:

MER	Mercator
NPS	North Polar Stereographic
SPS	South Polar Stereographic
LCC	Northern Hemisphere Lambert Conic Conformal
SCC	Southern Hemisphere Lambert Conic Conformal
CED	Cylindrical Equidistant
MCD	Modified Cylindrical Equidistant
UTM	Universal Transverse Mercator
NOR	North Orthographic
SOR	South Orthographic

The following full map projections may also be specified:

MER	Mercator
CED	Cylindrical Equidistant
MCD	Modified Cylindrical Equidistant
STR	Polar Stereographic
AED	Azimuthal equidistant
ORT	Orthographic
LEA	Lambert equal area
GNO	Gnomonic
LCC	Northern Hemisphere Lambert Conic Conformal
SCC	Southern Hemisphere Lambert Conic Conformal
UTM	Universal Transverse Mercator
TVM	Transverse Mercator

There are two satellite projections available:

AOI	AOIPS/2 navigation
NPG	Naval Postgraduate School navigation

The graph projections are:

POL	polar coordinates
LIN	linear x and y
LOG	linear x, logarithmic y
KAP	linear x, y ** KAPPA (KAPPA = 2/7)

GG_SMAP (PROJ, GAREA, PROCUR, GARCUR, IRET)

Input parameters:

PROJ	CHAR*	Map projection
GAREA	CHAR*	Graphics area

Output parameters:

IRET	INTEGER	Return code 0 = normal return -2 = invalid graphics area -5 = invalid projection

8.12 GG_SNPG - DEFINE NPGS SATELLITE NAV

This subroutine sets the satellite navigation for a Naval Postgraduate School image.

GG_SNPG (GAREA, IRET)

Input parameters:

GAREA	CHAR*	Image name

Output parameters:

IRET	INTEGER	Return code 0 = normal return -5 = invalid projection

8.13 GG_WSTR - WRITE TITLE

This subroutine writes a string on a graphics plot. The string will be centered on the line specified. If LINE = 0, the string will be written one line from the bottom of the plot.

GG_WSTR (STRING, LINE, IRET)

Input parameters:

STRING	CHAR*	String to be written
LINE	INTEGER	Line number <0 = lines from bottom 0 = bottom line >0 = lines from top

Output parameters:

IRET	INTEGER	Return code 0 = normal return

CHAPTER 9

GRID SUPPORT (GR) LIBRARY

GR_ALGN	Align grid corners
GR_AXLV	Compute axis labels
GR_CLVL	Select contour levels
GR_CMPV	Compute contour levels
GR_CVAL	Select contour interval
GR_FILE	Open grid file for graphics
GR_FIXA	Fix area
GR_GALM	Find grid subset area
GR_GTIM	Process input time
GR_INTP	Interpolate grid data
GR_LEVL	Process input level
GR_LIST	List grids in file
GR_LTLN	Get lat/lon at grid points
GR_MBAN	Make Barnes analysis block
GR_MNAV	Make navigation block
GR_OPEN	Open grid file
GR_PACK	Decode grid packing info
GR_PLIN	Get points for cross section
GR_PLOC	Get input for grid point
GR_RBAN	Read Barnes analysis block
GR_RNAV	Read navigation block
GR_ROBS	Read grid relative winds
GR_SCAL	Compute grid scaling
GR_SNAV	Set navigation in GEMPLT
GR_STAT	Compute grid statistics
GR_WOBS	Get observed winds
GR_WTRM	Write grid identifier to terminal

Grid Support (GR) Library Summary

The grid support library subroutines allow manipulation of information in GEMPAK grid files. Subroutines are available to create and decode analysis and navigation block information. Also included are subroutines to open grid files, set the grid navigation in GEMPLT, check grid boundaries, and define contour levels.

GR_LIST is available to list all the grids in a file. In the past, this subroutine would list grids based upon partial grid specifications. Currently, it lists all the grids in a file.

Error codes:

[GR 3]	User typed EXIT.
[GR 2]	Note: data have been internally rescaled.
[GR -1]	Invalid input time.
[GR -2]	Invalid input level.
[GR -3]	Wind components not found.
[GR -4]	File ... cannot be opened.
[GR -5]	Invalid grid spacing.
[GR -6]	Invalid navigation block.
[GR -7]	Error defining grid navigation in GEMPLT.
[GR -8]	Invalid data range.
[GR -9]	Invalid grid or grid subset size.
[GR -10]	Invalid analysis type.
[GR -11]	Number of output LUNs must be positive.
[GR -12]	INPUT ... for grid point is invalid.
[GR -13]	Error in getting KX and KY from grid common block.
[GR -14]	Data could not be scaled.
[GR -15]	START and STOP values for axis are missing.
[GR -16]	Invalid input for GPACK. No packing will be done.
[GR -17]	Axis labelling interval cannot be determined.
[GR -18]	Endpoints are too close together.

GRID SUPPORT (GR) LIBRARY

GR Library Calls

GR_ALGN	(grdin, deltax, deltay, / grdout, kx, ky, iret)
GR_AXLV	(dmin, dmax, start, stop, rint, stradj, stpadj, / v, nv, iret)
GR_CLVL	(maxlvl, cmin, cmax, cint, dmin, dmax, / nlvl, clvl, iret)
GR_CMPV	(rmin, rmax, rint, maxlvl, / nlvl, clvl, iret)
GR_CVAL	(rmin, rmax, / rint, iret)
GR_FILE	(gdfile, wrtflg, / gdcur, igdfln, / lasttm, maxgrd, iret)
GR_FIXA	(igdfln, area, / areout, iret)
GR_GALM	(kx, ky, / imin, jmin, imax, jmax, iret)
GR_GTIM	(gdattm, firstm, lasttm, / gdtim1, gdtim2, iret)
GR_INTP	(inttyp, gx, gy, npts, kx, ky, grid, / sdint, iret)
GR_LEVL	(glevel, / level1, level2, iret)
GR_LIST	(nlun, luns, igdfln, mesage, / answer, iret)
GR_LTLN	(kx, ky, / rlat, rlon, iret)
GR_MBAN	(deltan, deltax, deltay, gbnds, ebnds, dbnds, / anlblk, iret)
GR_MNAV	(proj, kx, ky, rlat1, rlon1, rlat2, rlon2, angl1, angl2, angl3, angflg, / rnvblk, iret)
GR_OPEN	(gdfile, wrtflg, / gdcur, igdfln, / lasttm, anl, rnav, numgrd, maxgrd, newfil, iret)
GR_PACK	(gpack, / ipktyp, nbits, iret)
GR_PLIN	(endpts, / npts, rgx, rgy, rlat, rlon, iret)
GR_PLOC	(gpoint, / rgx, rgy, rlat, rlon, iret)
GR_RBAN	(anlblk, / deltan, deltax, deltay, gbnds, ebnds, dbnds, iextnd, iret)
GR_RNAV	(rnvblk, / proj, kx, ky, iret)

GR_ROBS (iflno, gdtime, level, ivcord, / grid1, grid2, igx, igy, iret)

GR_SCAL (cscale, kx, ky, imin, jmin, imax, jmax, / grid, / iscale, rmin, rmax, iret)

GR_SNAV (navsz, rnvblk, / iret)

GR_STAT (z, kx, ky, imin, jmin, imax, jmax, / rmin, rmax, ravg, rdev, iret)

GR_WOBS (iflno, gdtime, level, ivcord, / grid1, grid2, wcmp, wmks, wparm, igx, igy, iret)

GR_WTRM (lun, title, ignum, gdattm, level, ivcord, parm, / iret)

9.1 GR_ALGN - ALIGN GRID CORNERS

This subroutine aligns a grid on grid points. The lower left corner specified in the input grid corners is moved to the left and down if necessary. The input and output grid corners are arrays ordered as follows: lower left lat, lower left lon, upper right lat, upper right lon.

```
GR_ALGN ( GRDIN, DELTAX, DELTAY, GRDOUT, KX, KY, IRET )
```

Input parameters:

GRDIN (4)	REAL	Input grid corners
DELTAX	REAL	X grid spacing
DELTAY	REAL	Y grid spacing

Output parameters:

GRDOUT (4)	REAL	Aligned grid corners
KX	INTEGER	Number of points in x dir
KY	INTEGER	Number of points in y dir
IRET	INTEGER	Return code 0 = normal return -5 = invalid grid spacing

9.2 GR_AXLV - COMPUTE AXIS LABELS

This subroutine defines axis label values given the data range, the axis range and labelling interval, if it is defined. A suitable label interval is determined automatically if it is missing.

GR_AXLV (DMIN, DMAX, START, STOP, RINT, STRADJ, STPADJ, V, NV,
IRET)

Input parameters:

DMIN	REAL	Minimum data value
DMAX	REAL	Maximum data value
START	REAL	Starting value on axis
STOP	REAL	Stopping value on axis
RINT	REAL	Labelling interval
STRADJ	LOGICAL	Flag to permit adjusting START
STPADJ	LOGICAL	Flag to permit adjusting STOP

Output parameters:

V (NV)	REAL	Array of label values
NV	INTEGER	Number of label values
IRET	INTEGER	Return code 0 = normal return -15 = START or STOP missing -17 = Scaling cannot be done

9.3 GR_CLVL - SELECT CONTOUR LEVELS

This subroutine selects contour levels given the range of data values in the grid subset area input for the contour interval and the minimum and maximum grid values. If the minimum or maximum input value is missing, the data value will be used. If the contour interval is non-positive, a contour interval producing five to ten contours will be selected.

GR_CLVL (MAXLVL, CMIN, CMAX, CINT, DMIN, DMAX, NLVL, CLVL, IRET)

Input parameters:

MAXLVL	INTEGER	Max number of contour levels
CMIN	REAL	Minimum contour value
CMAX	REAL	Maximum contour value
CINT	REAL	Contour interval
DMIN	REAL	Minimum data value
DMAX	REAL	Maximum data value

Output parameters:

NLVL	INTEGER	Number of contour levels
CLVL (NLVL)	REAL	Contour levels
IRET	INTEGER	Return code 0 = normal return -8 = invalid data range

9.4 GR_CMPV - COMPUTE CONTOUR LEVELS

This subroutine defines contour levels, given the data range and the contour interval.

GR_CMPV (RMIN, RMAX, RINT, MAXLVL, NLVL, CLVL, IRET)

Input parameters:

RMIN	REAL	Minimum value
RMAX	REAL	Maximum value
RINT	REAL	Contour interval
MAXLVL	INTEGER	Max number of contour levels

Output parameters:

NLVL	INTEGER	Number of contour levels
CLVL (NLVL)	REAL	Contour levels
IRET	INTEGER	Return code 0 = normal return -8 = invalid range

9.5 GR_CVAL - SELECT CONTOUR INTERVAL

This subroutine selects a contour interval, given minimum and maximum data values. The selected interval will generate five to ten contour levels.

GR_CVAL (RMIN, RMAX, RINT, IRET)

Input parameters:

RMIN	REAL	Minimum data value
RMAX	REAL	Maximum data value

Output parameters:

RINT	REAL	Contour interval
IRET	INTEGER	Return code 0 = normal return -8 = invalid data range

9.6 GR_FILE - OPEN GRID FILE FOR GRAPHICS

This subroutine opens a grid file. The input file name is first compared to the name of the current open grid file. If it is a new file, the old file is closed and the new file is opened. If the new open is successful, GDCUR is updated. This subroutine also sets the grid navigation in GEMPLT and initializes the grid diagnostics package by calling DG_INIT.

Note that the grid diagnostics subroutines now allow more than one open file. In order to use this feature, DG_OFIL should be used to open grid files.

GR_FILE (GDFILE, WRTFLG, GDCUR, IGDFLN, LASTTM, MAXGRD, IRET)

Input parameters:

GDFILE	CHAR*	File name input by user
WRTFLG	LOGICAL	Write access flag

Input and output parameters:

GDCUR	CHAR*	Current file name
IGDFLN	INTEGER	Grid file number

Output parameters:

LASTTM	CHAR*	Last time in file
MAXGRD	INTEGER	Maximum number of grids
IRET	INTEGER	Return code 0 = normal return -4 = grid file not opened -6 = grid navigation error

9.7 GR_FIXA - FIX AREA

This subroutine takes AREA and replaces GRID or DSET with the grid area, EXTEND with the extend area, and DATA with the data area. GRID or DSET is obtained from the navigation block; EXTEND and DATA are obtained from the analysis block.

GR_FIXA (IGDFLN, AREA, AREOUT, IRET)

Input parameters:

IGDFLN	INTEGER	Grid file number
AREA	CHAR*	Area

Output parameters:

AREOUT	CHAR*	New area
IRET	INTEGER	Return code 0 = normal return

9.8 GR_GALM - FIND GRID SUBSET AREA

This subroutine finds the boundaries of a subgrid which covers the graphics area.

GR_GALM (KX, KY, IMIN, JMIN, IMAX, JMAX, IRET)

Input parameters:

KX	INTEGER	Number of grid points in x dir
KY	INTEGER	Number of grid points in y dir

Output parameters:

IMIN	INTEGER	Minimum x value in area
JMIN	INTEGER	Minimum y value in area
IMAX	INTEGER	Maximum x value in area
JMAX	INTEGER	Maximum y value in area
IRET	INTEGER	Return code 0 = normal return -9 = invalid subgrid

9.9 GR_GTIM - PROCESS INPUT TIME

This subroutine changes the user input for grid time into two GEMPAK times. These two times are separated with a colon (:) and indicate the two times used to compute the grid function.

GR_GTIM (GDATTM, FIRSTM, LASTTM, GDTIM1, GDTIM2, IRET)

Input parameters:

GDATTM	CHAR*	Grid time input
FIRSTM	CHAR*	First time in grid file
LASTTM	CHAR*	Last time in grid file

Output parameters:

GDTIM1	CHAR*	First input time
GDTIM2	CHAR*	Second input time
IRET	INTEGER	Return code 0 = normal return -1 = invalid input time

9.10 GR_INTP - INTERPOLATE GRID DATA

This subroutine interpolates data from a grid to a set of points defined in GX, GY. Bilinear interpolation is the only interpolation type implemented.

GDPINT (INTTYP, GX, GY, NPTS, KX, KY, GRID, SDINT, IRET)

Input parameters:

INTTYP	INTEGER	Interpolation type
GX (NPTS)	REAL	Grid x coordinates
GY (NPTS)	REAL	Grid y coordinates
NPTS	INTEGER	Number of coordinates
KX	INTEGER	Number of x grid points
KY	INTEGER	Number of y grid points
GRID (KX,KY)	REAL	Grid data

Output parameters:

SDINT (npts)	REAL	Interpolated data
IRET	INTEGER	Return code 0 = normal return

9.11 GR_LEVL - PROCESS INPUT LEVEL

This subroutine changes the user input for grid level into two integers which represent the layer requested. If no value or invalid values are entered, the output level is set to -1. LIST is no longer an option in this subroutine.

GR_LEVL (GLEVEL, LEVEL1, LEVEL2, IRET)

Input parameters:

GLEVEL	CHAR*	Grid level input

Output parameters:

LEVEL1	INTEGER	First level of layer
LEVEL2	INTEGER	Second level of layer
IRET	INTEGER	Return code 0 = normal return -2 = invalid input level

9.12 GR_LIST - LIST GRIDS IN FILE

This subroutine lists all the grids in a grid file and prompts the user for input. The input will be returned in ANSWER. It is no longer possible to list only selected grids. The list may be sent to as many as four output units.

GR_LIST (NLUN, LUNS, IGDFLN, MESAGE, ANSWER, IRET)

Input parameters:

NLUN	INTEGER	Number of output units
LUNS (4)	INTEGER	Logical output unit numbers
IGDFLN	INTEGER	Grid file number
MESAGE	CHAR*	Message to write

Output parameters:

ANSWER	CHAR*	User input
IRET	INTEGER	Return code 3 = user entered EXIT 0 = normal return -11 = less than 1 output LUN

9.13 GR_LTLN - GET LAT/LON AT GRID POINTS

This subroutine computes the latitude and longitude at each grid point. The grid must be defined in GEMPLT before this subroutine is called.

```
GR_LTLN  ( KX, KY, RLAT, RLON, IRET )

Input parameters:
   KX                  INTEGER          Number of points in x dir
   KY                  INTEGER          Number of points in y dir

Output parameters:
   RLAT (KX,KY)        REAL             Latitudes in degrees
   RLON (KX,KY)        REAL             Longitudes in degrees
   IRET                INTEGER          Return code
                                          0 = normal return
                                         -6 = grid projection error
```

9.14 GR_MBAN - MAKE BARNES ANALYSIS BLOCK

This subroutine makes a Barnes analysis block. The analysis block generated is 128 words long. All the bounds must be entered in the order: lower left latitude; lower left longitude; upper right latitude; upper right longitude.

GR_MBAN (DELTAN, DELTAX, DELTAY, GBNDS, EBNDS, DBNDS, ANLBLK, IRET)

Input parameters:

DELTAN	REAL	Station spacing
DELTAX	REAL	Grid spacing in x dir
DELTAY	REAL	Grid spacing in y dir
GBNDS (4)	REAL	Grid area bounds
EBNDS (4)	REAL	Extended area bounds
DBNDS (4)	REAL	Data area bounds

Output parameters:

ANLBLK (128)	REAL	Analysis block
IRET	INTEGER	Return code 0 = normal return

9.15 GR_MNAV - MAKE NAVIGATION BLOCK

This subroutine makes a navigation block for a grid file. The projection may be any simple, full or graph projection. If ANGFLG is set, the projection must be a full map projection. Otherwise, a simple map projection will be defined.

```
GR_MNAV ( PROJ, KX, KY, RLAT1, RLON1, RLAT2, RLON2, ANGL1,
          ANGL2, ANGL3, ANGFLG, RNVBLK, IRET )
```

Input parameters:

PROJ	CHAR*	Projection name
KX	INTEGER	Number of x grid points
KY	INTEGER	Number of y grid points
RLAT1	REAL	Lower left latitude/x
RLON1	REAL	Lower left longitude/y
RLAT2	REAL	Upper right latitude/x
RLON2	REAL	Upper right longitude/y
ANGL1	REAL	Projection angle 1
ANGL2	REAL	Projection angle 2
ANGL3	REAL	Projection angle 3
ANGFLG	LOGICAL	Full projection flag

Output parameters:

RNVBLK (256)	REAL	Navigation block
IRET	INTEGER	Return code 0 = normal return

9.16 GR_OPEN - OPEN GRID FILE

This subroutine opens a grid file. The input file name is first compared to the name of the current open grid file. If it is a new file, the old file is closed and the new file is opened. If the new open is successful, GDCUR is updated.

Note that this subroutine does not set the navigation information in GEMPLT or initialize the DG package.

```
GR_OPEN  ( GDFILE, WRTFLG, GDCUR, IGDFLN, LASTTM, ANL, RNAV,
           NUMGRD, MAXGRD, NEWFIL, IRET )
```

Input parameters:

GDFILE	CHAR*	Grid file name
WRTFLG	LOGICAL	Write access flag

Input and output parameters:

GDCUR	CHAR*	Current file name
IGDFLN	INTEGER	Grid file number

Output parameters:

LASTTM	CHAR*	Last time in file
ANL (*)	REAL	Analysis block
RNAV (*)	REAL	Navigation block
NUMGRD	INTEGER	Number of grids in file
MAXGRD	INTEGER	Maximum number of grids
NEWFIL	LOGICAL	New file flag
IRET	INTEGER	Return code 0 = normal return -4 = file open error

9.17 GR_PACK - DECODE GRID PACKING INFO

This subroutine decodes the user input for grid packing into the number of bits / precision and packing type. The valid packing types are GRIB, DEC and DIF. If the packing type is DEC, NBITS is the precision; otherwise, NBITS is the number of bits.

GR_PACK (GPACK, IPKTYP, NBITS, IRET)

Input parameters:

GPACK	CHAR*	Number of bits / packing type

Output parameters:

IPKTYP	INTEGER	GEMPAK packing type
NBITS	INTEGER	Number of bits
IRET	INTEGER	Return code
		0 = normal return

9.18 GR_PLIN - GET POINTS FOR CROSS SECTION

This subroutine translates the user input for the end points of a cross-section line through a grid into an array of locations along the line segment. The locations in the output array are evenly spaced, with the spacing being approximately the grid spacing.

GR_PLIN (ENDPTS, NPTS, RGX, RGY, RLAT, RLON, IRET)

Input parameters:

ENDPTS	CHAR*	User input for end points

Output parameters:

NPTS	REAL	Number of points along line
RGX (NPTS)	REAL	X grid point
RGY (NPTS)	REAL	Y grid point
RLAT (NPTS)	REAL	Latitude
RLON (NPTS)	REAL	Longitude
IRET	INTEGER	Return code 0 = normal return -12 = invalid grid point -18 = endpoints too close

9.19 GR_PLOC - GET INPUT FOR GRID POINT

This subroutine translates the user input for a grid point into an actual grid point, x and y coordinates, and latitude and longitude.

GR_PLOC (GPOINT, RGX, RGY, RLAT, RLON, IRET)

Input parameters:

GPOINT	CHAR*	User input for grid point

Output parameters:

RGX	REAL	X grid point
RGY	REAL	Y grid point
RLAT	REAL	Latitude
RLON	REAL	Longitude
IRET	INTEGER	Return code 0 = normal return -12 = invalid grid point -13 = error in getting KX, KY

9.20 GR_RBAN - READ BARNES ANALYSIS BLOCK

This subroutine reads a Barnes analysis block. All the bounds are returned in the order: lower left latitude; lower left longitude; upper right latitude; upper right longitude.

```
GR_RBAN ( ANLBLK, DELTAN, DELTAX, DELTAY, GBNDS, EBNDS, DBNDS,
          IEXTND, IRET )
```

Input parameters:

ANLBLK (128)	REAL	Analysis block

Output parameters:

DELTAN	REAL	Station spacing
DELTAX	REAL	Grid spacing in x dir
DELTAY	REAL	Grid spacing in y dir
GBNDS (4)	REAL	Grid area bounds
EBNDS (4)	REAL	Extend area bounds
DBNDS (4)	REAL	Data area bounds
IEXTND (4)	INTEGER	Extend grid points
IRET	INTEGER	Return code 0 = normal return -10 = invalid analysis block

9.21 GR_RNAV - READ NAVIGATION BLOCK

This subroutine gets the projection and grid size from a grid navigation block.

GR_RNAV (RNVBLK, PROJ, KX, KY, IRET)

Input parameters:

RNVBLK (256)	REAL	Navigation block

Output parameters:

PROJ	CHAR*	Projection name
KX	INTEGER	Number of points in x dir
KY	INTEGER	Number of points in y dir
IRET	INTEGER	Return code 0 = normal return -6 = invalid navigation

9.22 GR_ROBS - READ GRID RELATIVE WINDS

This subroutine retrieves grid relative observed wind components from a grid file. The grid components must be stored as UREL and VREL.

```
GR_ROBS  ( IFLNO, GDTIME, LEVEL, IVCORD, GRID1, GRID2, IGX, IGY,
           IRET )
```

Input parameters:

IFLNO	INTEGER	Grid file number
GDTIME (2)	CHAR*	Grid time
LEVEL (2)	INTEGER	Grid level
IVCORD	INTEGER	Grid vertical coordinate

Output parameters:

GRID1 (*)	REAL	U-component grid
GRID2 (*)	REAL	V-component grid
IGX	INTEGER	Number of points in x dir
IGY	INTEGER	Number of points in y dir
IRET	INTEGER	Return code 0 = normal return -3 = wind unavailable

9.23 GR_SCAL - COMPUTE GRID SCALING

This subroutine computes the scaling term to be used for scaling grid data. If CSCALE contains a number, it will be used as a scaling factor. If CSCALE is missing, undefined or greater than 20 in absolute value, an appropriate scaling factor will be computed. The grid data are multiplied by 10 ** ISCALE. If the data are too small to be scaled with ISCALE = 20, ISCALE is set to IMISSD and IRET = -14.

```
GRSCAL  ( CSCALE, KX, KY, IMIN, JMIN, IMAX, JMAX, GRID, ISCALE,
          RMIN, RMAX, IRET )
```

Input parameters:

CSCALE	CHAR*	Input scale factor
KX	INTEGER	Number of grid points in x dir
KY	INTEGER	Number of grid points in y dir
IMIN	INTEGER	Minimum x grid point
JMIN	INTEGER	Minimum y grid point
IMAX	INTEGER	Maximum x grid point
JMAX	INTEGER	Maximum y grid point

Input and output parameters:

GRID (KX,KY)	REAL	Grid of data to be scaled

Output parameters:

ISCALE	INTEGER	Scale factor
RMIN	REAL	Data minimum
RMAX	REAL	Data maximum
IRET	INTEGER	Return code 0 = normal return -8 = no data in range -9 = invalid subset range -14 = scaling not possible

9.24 GR_SNAV - SET NAVIGATION IN GEMPLT

This subroutine sets up a grid coordinate system in GEMPLT. The navigation block should be sent as it was received from the grid file open subroutine. Note that the graphics projection and mode must be defined before GR_SNAV is called. This subroutine will fail if the grid mode is not the same as the current GEMPLT mode.

GR_SNAV (NAVSZ, RNVBLK, IRET)

Input parameters:

NAVSZ	INTEGER	Length of navigation block
RNVBLK (NAVSZ)	REAL	Navigation block

Output parameters:

IRET	INTEGER	Return code 0 = normal return -6 = invalid navigation type -7 = GEMPLT error

9.25 GR_STAT - COMPUTE GRID STATISTICS

This subroutine computes grid statistics.

GR_STAT (Z, KX, KY, IMIN, JMIN, IMAX, JMAX, RMIN, RMAX,
RAVG, RDEV, IRET)

Input parameters:

Z (KX,KY)	REAL	Data array
KX	INTEGER	Number of points in x dir
KY	INTEGER	Number of points in y dir
IMIN	INTEGER	Lower left corner of subgrid
JMIN	INTEGER	Lower left corner of subgrid
IMAX	INTEGER	Upper right corner of subgrid
JMAX	INTEGER	Upper right corner of subgrid

Output parameters:

RMIN	REAL	Minimum data value
RMAX	REAL	Maximum data value
RAVG	REAL	Average data value
RDEV	REAL	Standard deviation
IRET	INTEGER	Return code 0 = normal return -8 = no data in range -9 = invalid subset area

9.26 GR_WOBS - GET OBSERVED WINDS

This subroutine retrieves the observed wind components from a grid file. The grid file is searched for the following parameter names:

```
'UWND'  and  'VWND'
'UKNT'  and  'VKNT'
'DRCT'  and  'SPED'
'DRCT'  and  'SKNT'
```

```
GR_WOBS  ( IFLNO, GDTIME, LEVEL, IVCORD, GRID1, GRID2, WCMP,
           WMKS, WPARM, IGX, IGY, IRET )
```

Input parameters:

IFLNO	INTEGER	Grid file number
GDTIME (2)	CHAR*	Grid time
LEVEL (2)	INTEGER	Grid level
IVCORD	INTEGER	Grid vertical coordinate

Output parameters:

GRID1 (*)	REAL	First grid
GRID2 (*)	REAL	Second grid
WCMP	LOGICAL	Component type true = u,v false = speed,direction
WMKS	LOGICAL	MKS units flag
WPARM	CHAR*	Wind components concatenated
IGX	INTEGER	Number of points in x dir
IGY	INTEGER	Number of points in y dir
IRET	INTEGER	Return code 0 = normal return -3 = wind unavailable

9.27 GR_WTRM - WRITE GRID IDENTIFIER TO TERMINAL

This subroutine writes a grid identifier to the specified logical unit using a standard format. TITLE is set to indicate that the title line:

NUM TIME1 TIME2 LEVEL1 LEVEL2 VCORD PARM

is to be written first. If IGNUM is not positive, the grid number will not be written and will not be included in the title.

GR_WTRM (LUN, TITLE, IGNUM, GDATTM, LEVEL, IVCORD, PARM, IRET)

Input parameters:

LUN	INTEGER	Logical unit for write
TITLE	LOGICAL	Flag to write title
IGNUM	INTEGER	Grid number
GDATTM (2)	CHAR*20	GEMPAK time
LEVEL (2)	INTEGER	Vertical levels
IVCORD	INTEGER	Vertical coordinate
PARM	CHAR*12	Parameter name

Output parameters:

IRET	INTEGER	Return code 0 = normal return

CHAPTER 10

INPUT (IN) LIBRARY

IN_AXIS	Process AXIS
IN_BDTA	GEMPAK BLOCKDATA
IN_CINT	Process CINT
IN_COLR	Process COLOR
IN_LINE	Process LINE
IN_MARK	Process MARKER
IN_MRGD	Process MRGDAT
IN_OUTT	Process OUTPUT
IN_PARM	Process PARMS
IN_PRMC	Process PARMS and conditions
IN_PRMF	Process packing info
IN_PTYP	Process PTYPE
IN_SKYC	Decode sky coverage symbol
IN_TAXS	Process TAXIS
IN_TEXT	Process TEXT
IN_TITL	Process TITLE
IN_WIND	Process WIND
IN_WSYM	Decode weather symbol

Input Parameter (IN) Library Summary

The input parameter library is used to decode user input for standard GEMPAK variables.

ERROR MESSAGES:

[IN +2]	WARNING, no output has been requested.
[IN +1]	START or STOP not specified for axis.
[IN -1]	Error opening OUTPUT files.
[IN -2]	... for AXIS is insufficient or invalid.
[IN -3]	INPUT is invalid for CINT.
[IN -4]	Grid data for contouring is all missing or constant.
[IN -5]	Mandatory levels requested, but coordinate is not p.
[IN -6]	Axis limits are missing or indeterminant.
[IN -7]	Range for time axis is zero.
[IN -8]	Time axis increment results in too many blocks.

IN Library Calls

IN_AXIS	(axis, ivcrd, skewt, parm, dmin, dmax, ilfdef, igfdef, itfdef, / start, stop, values, nval, ilbfrq, iglfrq, itmfrq, iret)
IN_BDTA	(/ iret)
IN_CINT	(cint, grid, npts, / gmin, gmax, / cval, nv, iret)
IN_COLR	(colors, nexp, / icolor, iret)
IN_LINE	(line, values, nexp, / icolor, itype, iwidth, ilabel, iret)
IN_MARK	(marker, / mkcolr, iret)
IN_MRGD	(mrgdat, / mrgflg, ipttyp, iret)
IN_OUTT	(output, name, / lun, nlun, devs, iret)
IN_PARM	(nexp, parms, / prmlst, nparm, iret)
IN_PRMC	(nexp, parms, / prmlst, prmcnd, nparm, iret)
IN_PRMF	(prmfil, / nparm, parms, iscale, iofset, ibits, pkflg, iret)
IN_PTYP	(ptype, / iyaxis, ratio, rmargn, iret)
IN_SKYC	(skysym, / iret)
IN_TAXS	(taxis, maxlbl, npts, timfnd, / x, xstrt, xstop, xtlbl, ctlbl, nxlbl, xmndst, ilbfrq, iglfrq, itmfrq, iret)
IN_TEXT	(text, / iret)
IN_TITL	(title, idlin, / icttl, linttl, ttlstr, iret)
IN_WIND	(wind, / wintyp, winuni, iwnclr, iret)
IN_WSYM	(wsym, / iret)

10.1 IN_AXIS - PROCESS AXIS

This subroutine processes an axis variable. The start and stop values along with an array of values are returned. The frequencies with respect to the elements in the array of values for the plotting of labels, grid lines and tick marks are also returned. If any of these frequency values is missing, the corresponding output value is set to the input defaults. Plotting begins with the first element.

AXIS is expected to be of the form:

```
        start/stop/increment/labfrq;glnfrq;ticfrq
   or
        start/stop/value1;value2;...;valueN/labfrq;glnfrq;ticfrq
```

In the latter case, the increment specification has been replaced with a list of values. Failure to specify START and STOP will result in default values determined on the basis of DMIN and DMAX, the vertical coordinate or the parameter. If increment = MAN, then the mandatory levels between START and STOP are returned. A positive increment will generate values divisible by the increment. A negative increment will generate values incremented from START using the absolute value of the increment. If the SKEWT flag is set, extra lines will be added on the lower end of the scale.

NOTE: Dimension VALUES to LLAXIS in the calling program.

```
IN_AXIS  ( AXIS, IVCRD, SKEWT, PARM, DMIN, DMAX, ILFDEF, IGFDEF,
           ITFDEF, START, STOP, VALUES, NVAL, ILBFRQ, IGLFRQ,
           ITMFRQ, IRET )
```

Input parameters:

AXIS	CHAR*	Input for axis
IVCRD	INTEGER	Vertical coordinate 0 = NONE 2 = THTA 1 = PRES 3 = HGHT
SKEWT	LOGICAL	Flag skewT plot T axis
PARM	CHAR*	Parameter name (optional)
DMIN	REAL	Data minimum
DMAX	REAL	Data maximum
ILFDEF	INTEGER	Default label frequency
IGFDEF	INTEGER	Default grid line frequency
ITFDEF	INTEGER	Default tick mark frequency

Output parameters:

START	REAL	Starting value for axis
STOP	REAL	Stopping value for axis
VALUES (NVAL)	REAL	Array of values

NVAL	INTEGER	Number of values
ILBFRQ	INTEGER	Label frequency
IGLFRQ	INTEGER	Grid line frequency
ITMFRQ	INTEGER	Tick mark frequency
IRET	INTEGER	Return code 0 = normal return -2 = incorrect specification -5 = MAN lvls not appropriate -6 = axis bounds are missing

10.2 IN_BDTA - GEMPAK BLOCKDATA

This subroutine serves as a BLOCKDATA statement, initializing variables in GEMPAK common blocks. This subroutine is called by IP_INIT. If a GEMPAK program does not call IP_INIT, it must call IN_BDTA directly.

IN_BDTA (IRET)

Output parameters:

IRET	INTEGER	Return code 0 = normal return

10.3 IN_CINT - PROCESS CINT

This subroutine processes the user contour specification. If GMIN or GMAX is set to the missing value, the values in the grid are used to compute GMIN and GMAX. GRID is not used if GMIN and GMAX are not missing. An array of values is returned.

CINT is expected to be of the form:

```
        increment/minimum/maximum
   or
          value1;value2;...;valueN
```

where the minimum and maximum give the range for the contours. If the minimum equals the maximum, a single contour with that value is assumed. In the latter specification, the specified contour levels are used, and the minimum and maximum are ignored.

IN_CINT (CINT, GRID, NPTS, GMIN, GMAX, CVAL, NV, IRET)

Input parameters:

CINT	CHAR*	Input for contours
GRID (NPTS)	REAL	Array of data
NPTS	INTEGER	Number of grid values

Input and output paramters:

GMIN	REAL	Minimum grid value
GMAX	REAL	Maximum grid value

Output parameters:

CVAL	REAL	List of contour levels
NV	INTEGER	Number of contour levels
IRET	INTEGER	Return code 0 = normal return -3 = invalid input -4 = constant grid

10.4 IN_COLR - PROCESS COLOR

This subroutine converts the input for the COLORS variable into a list of colors. If the number of colors is less than the number expected, the input colors will be repeated to fill the buffer. If COLORS is blank, the default is color 1.

The colors can now be queried or set by name. The color names corresponding to the color numbers can be listed by ending the color list with a ?. Color numbers can be set to specific colors by using the =. For example, 1=red;2=orange;3=blue;4;5? will set color number 1 to red, 2 to orange, 3 to blue and it will list the current color names for all color numbers.

IN_COLR (COLORS, NEXP, ICOLOR, IRET)

Input parameters:

COLORS	CHAR*	COLORS input
NEXP	INTEGER	Number of colors

Output parameters:

ICOLOR (NEXP)	INTEGER	Color number array
IRET	INTEGER	Return code 0 = normal return

10.5 IN_LINE - PROCESS LINE

This subroutine converts the input for the LINE variable into a list of colors, line types, line widths and line label flags. If the number of specifications is less than the number expected, the input sequence will be repeated to fill the buffer.

The LINE input must be of the form:

```
col1;col2;.../typ1;typ2.../wid1;wid2.../lab1;lab2...
```

In general, lines are turned off by specifying color = 0. 0 for line type or width will use a default value of 1. 0 for line label will suppress labelling.

Note that the colors can now be set or queried by name. See IN_COLR for details.

If the line type is set to a single negative number, negative values will have the line type specified and positive values will be solid (line type = 1). If the label is set to a single number, say n, then every nth value will be labelled.

```
IN_LINE ( LINE, VALUES, NEXP, ICOLOR, ITYPE, IWIDTH, ILABEL,
          IRET )
```

Input parameters:

LINE	CHAR*	LINE input
VALUES (NEXP)	REAL	Data values to draw and label
NEXP	INTEGER	Number expected

Output parameters:

ICOLOR (NEXP)	INTEGER	Color number array
ITYPE (NEXP)	INTEGER	Line type number array
IWIDTH (NEXP)	INTEGER	Line width number array
ILABEL (NEXP)	INTEGER	Line label number array
IRET	INTEGER	Return code 0 = normal return

10.6 IN_MARK - PROCESS MARKER

This subroutine decodes the marker string which is in the form:

color # / marker # / size / width / hw, sw flag

Note that the hw, sw flag can appear anywhere in the string.

The marker size is a real number which is a multiplier for the base marker size. If the size is 0.0, the current size will be used. If the marker color is 0, no marker will be drawn. If the marker color is blank, color number 1 will be used. If the marker number is missing or 0, the current marker number will be used. The marker type, size and width are set in this subroutine, while the color is returned so that the program may set it when actually plotting markers. The GEMPLT package must be initialized before this subroutine is called.

IN_MARK (MARKER, MKCOLR, IRET)

Input parameters:

MARKER	CHAR*	Marker input

Output parameters:

MKCOLR	INTEGER	Marker color
IRET	INTEGER	Return code 0 = normal return

10.7 IN_MRGD - PROCESS MRGDAT

This subroutine breaks the user input for MRGDAT into a flag indicating whether merged data are to be used and a type for unmerged data. The default for MRGFLG is true and for IPTTYP is 3.

IN_MRGD (MRGDAT, MRGFLG, IPTTYP, IRET)

Input parameters:

MRGDAT	CHAR*	User input for MRGDAT

Output parameters:

MRGFLG	LOGICAL	Merged file type
IPTTYP	INTEGER	Unmerged type 1 = man below 100 mb 2 = man & sig below 100 mb 3 = man & sig below & above
IRET	INTEGER	Return code 0 = normal return

10.8 IN_OUTT - PROCESS OUTPUT

This subroutine processes the OUTPUT variable. The requested output types are determined and corresponding logical unit numbers are returned. Output may be directed to the terminal, a file, or a printer. OUTPUT will be searched for 'T', 'P', and 'F' to determine which output devices are requested. If the output devices are followed by a slash and a string, the string will be used as the name of the output file. If file output is requested and no file name is specified, the file will be called NAME.FIL, where name is an input variable to this subroutine and should be the name of the executing program. If no valid devices are specified, output will be sent to the terminal. If the output request contains an 'N' before the slash, no output will be written.

IN_OUTT (OUTPUT, NAME, LUN, NLUN, DEVS, IRET)

Input parameters:

OUTPUT	CHAR*	Output variable
NAME	CHAR*	Program name

Output parameters:

LUN (NLUN)	INTEGER	Logical unit numbers
NLUN	INTEGER	Number of output devices
DEVS (NLUN)	CHAR*1	Device name (T,P,F)
IRET	INTEGER	Return code 0 = normal return -1 = error opening files

10.9 IN_PARM - PROCESS PARMS

This subroutine processes the input variable PARMS where the input string contains a list of parameter names separated by semicolons. All spaces are eliminated from the input string. If two consecutive semicolons are found, the parameter BLNK will be inserted.

IN_PARM (NEXP, PARMS, PRMLST, NPARM, IRET)

Input parameters:

NEXP	INTEGER	Maximum number of parameters
PARMS	CHAR*	Parameter string

Output parameters:

PRMLST (NPARM)	CHAR*	Parameter array
NPARM	INTEGER	Number of parameters
IRET	INTEGER	Return code
		0 = normal return

10.10 IN_PRMC - PROCESS PARMS AND CONDITIONS

This subroutine processes the input variable PARMS where the input string contains a list of parameter names separated with semicolons. All spaces are eliminated from the input string. If two consecutive semicolons are found, the parameter BLNK will be inserted. Any characters after the fourth character in the parameter name are returned as conditions. This subroutine is the same as IN_PARM except that the condition array is returned here.

IN_PRMC (NEXP, PARMS, PRMLST, PRMCND, NPARM, IRET)

Input parameters:

NEXP	INTEGER	Maximum number of parameters
PARMS	CHAR*	Parameter string

Output parameters:

PRMLST (NPARM)	CHAR*4	Parameter array
PRMCND (NPARM)	CHAR*	Parameter condition array
NPARM	INTEGER	Number of parameters
IRET	INTEGER	Return code 0 = normal return

10.11 IN_PRMF - PROCESS PACKING INFO

This subroutine receives the user input for parameter packing and returns the packing information. The input is either the name of a file containing the information or the information itself entered as follows:

```
PRM1/MIN1-MAX1-RES1;PRM2/MIN2-MAX2-RES2; ...
```

where PRMn is the parameter name, MINn is the minimum for PRMn, MAXn is the maximum for PRMn, and RESn is the resolution.

```
IN_PRMF ( PRMFIL, NPARM, PARMS, ISCALE, IOFSET, IBITS, PKFLG
          IRET )
```

Input parameters:

PRMFIL		CHAR*	Input packing information

Output parameters:

NPARM		INTEGER	Number of parameters
PARMS	(NPARM)	CHAR*	Parameter names
ISCALE	(NPARM)	INTEGER	Scaling for packing
IOFSET	(NPARM)	INTEGER	Offset for packing
IBITS	(NPARM)	INTEGER	Number of packing bits
PKFLG		LOGICAL	Packing flag
IRET		INTEGER	Return code 0 = normal return -9 = invalid packing info -10 = all parms must be packed

10.12 IN_PTYP - PROCESS PTYPE

This subroutine translates the variable PTYPE and returns values for the axis type, height-to-width ratio, and the margins. If the margins are not specified, -1. is returned.

IN_PTYP (PTYPE, IYAXIS, RATIO, RMARGN, IRET)

Input parameters:

PTYPE	CHAR*	Y axis input

Output parameters:

IYAXIS	INTEGER	Y axis integer type
RATIO	REAL	Height-to-width ratio
RMARGN (4)	REAL	Margins
IRET	INTEGER	Return code 0 = normal return -7 = invalid axis type

10.13 IN_SKYC - DECODE SKY COVERAGE SYMBOL

This subroutine decodes the input for the sky coverage symbol. The variable has three parts separated by slashes. The first part contains the symbol size. The second part contains the symbol width. The third part contains the sky coverage symbol type. The decoded values are used to set the sky symbol defaults in GEMPLT.

IN_SKYC (SKYSYM, IRET)

Input parameters:

SKYSYM	CHAR*	Sky coverage symbol input

Output parameters:

IRET	INTEGER	Return code 0 = normal return

10.14 IN_TAXS - PROCESS TAXIS

This subroutine determines the values to use for the time axis in a time series program. TAXIS must be in the form:

```
tstart-tstop-tinc;lblfrq;glnfrq;ticfrq
```

where the last three are the frequencies for labels, grid lines and tick marks.

Defaults will be set for all values not supplied explicitly.

```
IN_TAXS  ( TAXIS, MAXLBL, NPTS, TIMFND, X, XSTRT, XSTOP, XTLBL,
           CTLBL, NXLBL, XMNDST, ILBFRQ, IGLFRQ, ITMFRQ, IRET )
```

Input parameters:

TAXIS	CHAR*	User input for T axis
MAXLBL	INTEGER	Maximum number of labels
NPTS	INTEGER	Number of times
TIMFND (NPTS)	CHAR*	GEMPAK times

Output parameters:

X (NPTS)	REAL	X positions of times in days
XSTRT	REAL	Left value of x
XSTOP	REAL	Right value of x
XTLBL (NXLBL)	REAL	X axis label positions
CTLBL (NXLBL)	CHAR*	X axis labels
NXLBL	INTEGER	Number of x axis labels
XMNDST	REAL	Min time separation in days
ILBFRQ	INTEGER	Label frequency
IGLFRQ	INTEGER	Grid line frequency
ITMFRQ	INTEGER	Tick mark frequency
IRET	INTEGER	Return code 0 = normal return -7 = time range is size zero -8 = too many labels

10.15 IN_TEXT - PROCESS TEXT

This subroutine decodes the text string which is in the form:

text size / text font / text width / hw,sw flag

Note that the hw,sw flag can appear anywhere in the string. The specified characteristics are set in GEMPLT.

If any parameter is not input, the current default will be used. The GEMPLT graphics package must be initialized before this subroutine is called.

IN_TEXT (TEXT, IRET)

Input parameters:

TEXT	CHAR*	Text input

Output parameters:

IRET	INTEGER	Return code 0 = normal return

10.16 IN_TITL - PROCESS TITLE

This subroutine converts the input for the TITLE variable into a title color, title line and title string. The inputs for TITLE are separated by slashes.

IN_TITL (TITLE, IDLIN, ICTTL, LINTTL, TTLSTR, IRET)

Input parameters:

TITLE	CHAR*	TITLE input
IDLIN	INTEGER	Default line

Output parameters:

ICTTL	INTEGER	Title color
LINTTL	INTEGER	Title line
TTLSTR	CHAR*	Title string
IRET	INTEGER	Return code
		0 = normal return

10.17 IN_WIND - PROCESS WIND

This subroutine decodes the input for WIND. The variable has two parts separated by a slash. The first part contains the wind type (B for barb, A for arrow), the wind units (M for meters/sec, K for knots) and the color number. There should be no slashes in this part. The second part contains the size, width, type of the arrow or barb, and the arrowhead size separated by slashes. The arrow/barb size is a multiple of the base size. Type 1 plots a circle or an arrowhead for calm winds. Type 2 does not plot anything for calm winds. The arrowhead size is a multiple of the base arrowhead size.

An example of the wind string is: BM/1.0/5/2

IN_WIND (WIND, WINTYP, WINUNI, IWNCLR, IRET)

Input parameters:

WIND	CHAR*1	Wind input

Output parameters:

WINTYP	CHAR*1	Wind type B = wind barb A = wind arrow
WINUNI	CHAR*	Wind units K = knots M = meters/second
IWNCLR	INTEGER	Wind color
IRET	INTEGER	Return code 0 = normal return

10.18 IN_WSYM - DECODE WEATHER SYMBOL

This subroutine decodes the input for the weather symbol. The variable has two parts each preceded by a *. The first part contains the weather symbol size and the second part contains the weather symbol width.

IN_WSYM (WSYM, IRET)

Input parameters:

WSYM	CHAR*	Weather symbol input

Output parameters:

IRET	INTEGER	Return code 0 = normal return

CHAPTER 11

TAE INPUT PARAMETER (IP) LIBRARY

IP_DYNM	Enter dynamic tutor
IP_EXIT	Exit from TAE
IP_IDNT	Program identification
IP_INIT	Initialize TAE
IP_LOG	Receive logical variable
IP_MFIL	Get map file name
IP_STR	Receive string variable
IP_ULOC	Update local TAE variable
IP_ULOG	Update global logical variable
IP_USTR	Update global string variable

TAE Input Parameter (IP) Library Summary

The TAE input parameter library provides an easy interface to the TAE subroutines. It makes the calls to the TAE required by standard GEMPAK programs.

If a program is to obtain any variables from the TAE, the subroutine IP_INIT must be called first. This subroutine initializes the TAE variable block. The program can then obtain input parameter values using the subroutines IP_STR or IP_LOG. GEMPAK no longer supports the return of integer or real parameters directly from the TAE. Also, arrays can no longer be received from the TAE.

After all parameters are checked for validity, the subroutine IP_USTR should be called for each parameter to update the global parameter value.

IP_DYNM is called by most GEMPAK programs to allow new parameters to be entered and the program to be executed again in a dynamic tutor.

IP_EXIT must be the last IP subroutine called before ending the program. It is used to update the global values.

Note that any program calling IP_INIT must include the global parameter $RESPOND on the REFGBL line of its PDF.

All errors from the TAE are printed when they are encountered by an IP subroutine.

If the user is not in the TAE or if there is an error initializing the TAE, non-TAE (NT) subroutines will be called. No changes need to be made in any applications program to use the non-TAE interface, provided that ALL TAE calls are made using the IP library subroutines.

ERROR MESSAGES FROM THE TAE:

[TAE 1800]	Invalid parameter name for parameter ...
[TAE 1801]	Invalid parameter type for parameter ...
[TAE 1805]	Duplicate parameter name for parameter ...
[TAE 1810]	Length of string insufficient for parameter ...
[TAE 1811]	TAE error.
[TAE 2102]	No parameter returned for parameter ...

ERROR MESSAGES FROM THE IP LIBRARY:

[IP -2] Error receiving parameter ....
[IP -3] Globals not updated.

IP Library Calls

```
IP_DYNM   ( / done, iret )
IP_EXIT   ( / iret )
IP_IDNT   ( / progrm, iret )
IP_INIT   ( / respnd, iret )
IP_LOG    ( pname, / logprm, iret )
IP_MFIL   ( / mapfil, iret )
IP_STR    ( pname, / parm, iret )
IP_ULOC   ( pname, parm, / iret )
IP_ULOG   ( pname, logvar, / iret )
IP_USTR   ( pname, parm, / iret )
```

11.1 IP_DYNM - ENTER DYNAMIC TUTOR

This subroutine takes the user into a dynamic tutor, allowing new values to be entered for the current program. If the global variable $RESPOND is set to NO or if the user enters EXIT, DONE will be set.

IP_DYNM (DONE, IRET)

Output parameters:

DONE	LOGICAL	Program exit flag
IRET	INTEGER	Return code 0 = normal return

11.2 IP_EXIT - EXIT FROM TAE

This subroutine performs the functions needed to exit from the TAE. Global variables which have been changed in the program are actually updated at this time. This subroutine must be called at the end of every program.

IP_EXIT (IRET)

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = globals not updated

11.3 IP_IDNT - PROGRAM IDENTIFICATION

This subroutine saves the name of the current program for the non-TAE dynamic tutor.

IP_IDNT (PROGRM, IRET)

Output parameters:

PROGRM	CHAR*	Program name
IRET	INTEGER	Return code 0 = normal return

11.4 IP_INIT - INITIALIZE TAE

This subroutine initializes the TAE. It reads the TAE block which contains both local and global variables and initializes a second variable block to be used for a dynamic tutor.

The variable RESPND returns the logical value of the global variable $RESPOND. Whenever the value of RESPND is FALSE, the program should not expect input from the user. If the user is executing the program in batch mode, the respond flag will also be set to false. GEMPAK programs currently do not use the value of RESPND. Instead, it is obtained each time it is required. The value is returned here to maintain compatibility with earlier versions.

If an error is encountered in initializing the TAE block, non-TAE (NT) code will be used.

IP_INIT (RESPND, IRET)

Output parameters:

RESPND	LOGICAL	Respond flag
IRET	INTEGER	Return code
		0 = normal return

11.5 IP_LOG - RECEIVE LOGICAL VARIABLE

This subroutine receives the value of a logical variable. A YES or NO entered in the TAE is converted to TRUE or FALSE. If the first letter of the input is not Y, the value is set to FALSE.

IP_LOG (PNAME, LOGPRM, IRET)

Input parameters:

PNAME	CHAR*	Name of variable

Output parameters:

LOGPRM (NPARM)	LOGICAL	Parameter values
IRET	INTEGER	Return code 0 = normal return -2 = parameter not received

11.6 IP_MFIL - GET MAP FILE NAME

This subroutine extracts the current map file name from $MAPFIL.

IP_MFIL (MAPFIL, IRET)

Output parameters:

MAPFIL	CHAR*	Map file name
IRET	INTEGER	Return code 0 = normal return -2 = parameter not received

11.7 IP_STR - RECEIVE STRING VARIABLE

This subroutine receives a string variable from the TAE.

IP_STR (PNAME, PARM, IRET)

Input parameters:

PNAME	CHAR*	Name of variable

Output parameters:

PARM	CHAR*	String
IRET	INTEGER	Return code 0 = normal return -2 = parameter not received

11.8 IP_ULOC - UPDATE LOCAL TAE VARIABLE

This subroutine saves a string variable in the local variable block used for dynamic tutors. It can be used to update variables within a program.

IP_ULOC (PNAME, PARM, IRET)

Input parameters:

PNAME	CHAR*	Variable name
PARM	CHAR*	Variable value

Output parameters:

IRET	INTEGER	Return code 0 = normal return

11.9 IP_ULOG - UPDATE GLOBAL LOGICAL VARIABLE

This subroutine updates a TAE global logical variable.

IP_ULOG (PNAME, LOGVAR, IRET)

Input parameters:

PNAME	CHAR*	Parameter name
LOGVAR	LOGICAL	Parameter value

Output parameters:

IRET	INTEGER	Return code 0 = normal return

11.10 IP_USTR - UPDATE GLOBAL STRING VARIABLE

This subroutine updates a TAE global string variable. The subroutine attempts to update a corresponding global variable. No error will be returned if there is no such global. Since all GEMPAK parameters now have a corresponding global value, each program should update all its variables.

IP_USTR (PNAME, PARM, IRET)

Input parameters:

PNAME	CHAR*	Parameter name
PARM	CHAR*	Parameter value

Output parameters:

IRET	INTEGER	Return code 0 = normal return

CHAPTER 12

LOCATION (LC) LIBRARY

LC_ABND	Decode subarea
LC_AREA	Process a subarea from AREA
LC_COUN	Check country name
LC_FLOC	Find location of point
LC_FSTN	Find station location
LC_GARE	Define GAREA
LC_INBN	Check lat/lon
LC_SARE	Define AREA
LC_SBND	Set lat/lon bounds
LC_UARE	Define new AREA

Location (LC) Library Summary

The GEMPAK location library provides subroutines for defining data subset and graphics areas in GEMPAK.

Areas containing subareas may now be defined using the subroutines LC_SARE and LC_UARE. Subareas must be separated by slashes (/). Each subarea is additive (+) or subtractive (-), depending on the first character following the slash, with + being the default. Additive subareas add stations to the list of valid stations; subtractive subareas eliminate stations which were previously valid.

Subareas may be specified in the following ways:

1. lat1;lon1;lat2;lon2
   This defines a latitude/longitude range where (lat1, lon1) is the lower left corner and (lat2, lon2) is the upper right corner.

   or

   #clat;clon;dlat;dlon
   This defines a latitude/longitude range where (clat-dlat, clon-dlon) is the lower left corner (clat+dlat, clon+dlon) is the upper right corner.

2. GEOG
   This is an abbreviation for a geographic area defined in the GEMPAK geographic table which includes abbreviations for states, provinces and countries, as well as other names. If #GEOG is entered, the user's geographic table, GEOG.TBL, will be searched. A * or - after the name may be used to reduce/expand the area.

3. STN
   This defines an area centered on a station found in the GEMPAK station table, which contains US and Canadian surface stations. A * or - after the name may be used to reduce/expand the area.

4. DSET
   This includes all the stations in the current dataset.

5. @ST
   This area includes those stations located in the state, province or country defined by ST. Only some countries

are recognized (US,CN,MX,CI,BW,AU); other countries may be specified using method 6.

6. @CN:C
   This area includes those stations located in the country defined by CN.

7. @STN1;STN2;...;STNn
   This area includes the stations listed, where STNn may be a station identifier or a station number.

8. SHDR:iloval:ihival
   This area defines a range of valid values for the station header, SHDR. For example, SELV:0:2000 specifies stations whose elevations are less than 2000 meters.

Graphics areas must be specified using methods 1 - 3.

Subroutines to set and check bounds, LC_SBND and LC_INBN, are included to maintain compatibility with earlier versions of GEMPAK.

ERROR MESSAGES:

[LC -1] ... is an invalid area name.
[LC -2] The geographic file cannot be opened.
[LC -3] The station file cannot be opened.
[LC -4] Area name ... is not in the table.
[LC -5] ... is an invalid graphics area.

LOCATION (LC) LIBRARY

LC Library Calls

LC_ABND (area, / iartyp, rlatll, rlonll, rlatur, rlonur, stn, nstn, stcn, iret)

LC_AREA (area, / rltln, stn, nstn, state, iartyp, iret)

LC_COUN (stcn, / cnflag, iret)

LC_FLOC (point, / rlat, rlon, iret)

LC_FSTN (stn, / rlat, rlon, iret)

LC_GARE (garea, / grltln, iret)

LC_INBN (rlat, rlon, / bound, iret)

LC_SARE (area, iflno, / stn, iret)

LC_SBND (rltln, / iret)

LC_UARE (area, newfil, iflno, / arecur, / stn, iret)

LOCATION (LC) LIBRARY

12.1 LC_ABND - DECODE SUBAREA

This subroutine translates a GEMPAK subarea. For area types 1-3, the latitude/longitude bounds are returned. For area type 2, STN contains the center station. For area type 6, STN contains the list of stations. For area types 5 and 7, the state or country is returned in STCN. Area types 2 and 3 may be followed by a number of * or - to contract or expand the region.

```
LC_ABND ( AREA, IARTYP, RLATLL, RLONLL, RLATUR, RLONUR, STN,
          NSTN, STCN, IRET )
```

Input parameters:

AREA	CHAR*	Area name

Output parameters:

IARTYP	INTEGER	Area type -1 = none 1 = area name 2 = center on station 3 = lat/lon bounds 4 = DSET 5 = @ST 6 = @STN1;...;STNN 7 = @CN:C
RLATLL	REAL	Lower left latitude
RLONLL	REAL	Lower left longitude
RLATUR	REAL	Upper right latitude
RLONUR	REAL	Upper right longitude
STN (NSTN)	CHAR*	Stations
NSTN	INTEGER	Number of stations
STCN	CHAR*	State/country
IRET	INTEGER	Return code 0 = area found -1 = invalid area name -3 = station file open error

12.2 LC_AREA - PROCESS A SUBAREA FROM AREA

This subroutine processes a single subarea from the input variable AREA. Information about the subarea is returned. No error messages are written.

LC_AREA (AREA, RLTLN, STN, NSTN, STATE, IARTYP, IRET)

Input parameters:

AREA	CHAR*	Area name

Output parameters:

RLTLN (4)	REAL	Latitude/longitude bounds
STN (NSTN)	CHAR*	Center station
NSTN	INTEGER	Number of stations
STATE	CHAR*	Center state of the area
IARTYP	INTEGER	Type of area -1 = none 1 = area name 2 = center on station 3 = latitude/longitude 4 = DSET 5 = @STATE 6 = @STN1;...;STNN 7 = @CN:C
IRET	INTEGER	Return code 0 = normal return -1 = invalid area name

12.3 LC_COUN - CHECK COUNTRY NAME

This subroutine checks STCN to see if it is a country abbreviation. The following countries are currently recognized:

US	United States	CN	Canada
MX	Mexico	BW	Bangladesh
AU	Australia	CI	China

Countries whose abbreviations will conflict with US state names should not be added to this list.

LC_COUN (STCN, CNFLAG, IRET)

Input parameters:

STCN	CHAR*	State / country abbreviation

Output parameters:

CNFLAG	LOGICAL	Country flag
IRET	INTEGER	Return code 0 = normal return

12.4 LC_FLOC - FIND LOCATION OF POINT

This subroutine translates a location into a latitude and longitude. The location may be entered in the following ways:

LAT;LON
LAT/LON
character station identifier
station number

The surface station table, the upper-air station table and the world station table will be searched for stations.

LC_FLOC (POINT, RLAT, RLON, IRET)

Input parameters:

POINT	CHAR*	Location

Output parameters:

RLAT	REAL	Latitude
RLON	REAL	Longitude
IRET	INTEGER	Return code 0 = normal return -4 = station not in table

12.5 LC_FSTN - FIND STATION LOCATION

This subroutine searches the station table file for a particular station and returns the latitude and longitude of the station.

The input parameter STN must be in upper case letters.

LC_FSTN (STN, RLAT, RLON, IRET)

Input parameters:

STN	CHAR*4	Station identifier

Output parameters:

RLAT	REAL	Station latitude
RLON	REAL	Station longitude
IRET	INTEGER	Return code 0 = normal return -3 = station table not opened -4 = station not in table

12.6 LC_GARE - DEFINE GAREA

This subroutine processes the input variable GAREA. Information about the type of area input is returned. Only those area types which specify a latitude/longitude range are valid.

LC_GARE (GAREA, GRLTLN, IRET)

Input parameters:

GAREA	CHAR*	Graphics area name

Output parameters:

GRLTLN (4)	REAL	Latitude/longitude bounds
IRET	INTEGER	Return code 0 = normal return -5 = invalid garea name

12.7 LC_INBN - CHECK LAT/LON

This subroutine checks a latitude / longitude pair to see if it is within the range specified by LC_SBND.

LC_INBN (RLAT, RLON, BOUND, IRET)

Input parameters:

RLAT	REAL	Latitude
RLON	REAL	Longitude

Output parameters:

BOUND	LOGICAL	Flag set if in bounds
IRET	INTEGER	Return code 0 = normal return

12.8 LC_SARE - DEFINE AREA

This subroutine sets the search criteria in a DM file using the value for AREA input by the user. The area may be composed of subareas which are separated by slashes (/). The DM file must be opened before this subroutine is called. If an invalid subarea is encountered, an error message is printed and an error is returned. If any subarea is centered on a station, that station is returned in STN. Note that any subroutine which defines a search, such as SF_SSTN, will eliminate the search set by this subroutine.

LC_SARE (AREA, IFLNO, STN, IRET)

Input parameters:

AREA	CHAR*	Area to be defined
IFLNO	INTEGER	File number for DM file

Output parameters:

STN	CHAR*	Center station name
IRET	INTEGER	Return code 0 = normal return -1 = invalid area name

12.9 LC_SBND - SET LAT/LON BOUNDS

This subroutine sets the latitude/longitude bounds of a geographic area. Once this subroutine has been called, the subroutine LC_INBN may be called to check whether a latitude/longitude location is within the specified range.

LC_SBND (RLTLN, IRET)

Input parameters:

RLTLN (4)	REAL	Lower left, upper right lat/lon

Output parameters:

IRET	INTEGER	Return code 0 = normal return

12.10 LC_UARE - DEFINE NEW AREA

This subroutine updates and processes the user input for AREA. It calls LC_SARE only if the area name has changed or a new file has been opened. This subroutine is useful if AREA is to be defined repeatedly. ARECUR is the current active area whose value is set in this subroutine and which should not be changed in the application program.

LC_UARE (AREA, NEWFIL, IFLNO, ARECUR, STN, IRET)

Input parameters:

AREA	CHAR*	Input for area
NEWFIL	LOGICAL	New file flag
IFLNO	INTEGER	File number

Input and output parameters:

ARECUR	CHAR*	Current area name

Output paramters:

STN	CHAR*	Station at center of area
IRET	INTEGER	Return code 0 = normal return -1 = invalid area name

CHAPTER 13

LEVEL (LV) LIBRARY

LV_CCRD	Get vertical coord name
LV_CORD	Get vertical coordinate
LV_DECD	Decode single level
LV_DUPL	Eliminate duplicate levels
LV_GRNG	Get levels from range
LV_INPT	Decode input for LEVEL and VCOORD
LV_MANL	Return mandatory levels
LV_SORT	Sort levels
LV_VASL	Return VAS levels

Level (LV) Library Summary

The LEVEL library processes user input for vertical level and vertical coordinate.

LV_INPT translates the user inputs for LEVELS and VCOORD into a list of levels. The input for LEVELS may be a list separated by semicolons. The following items may be included in the list:

a single level;
MAN for the mandatory levels below 100 mb;
VAS for the standard VAS levels;
a range of levels with an increment.

The following items are also valid, provided they are not part of a list:

ALL for all levels;
a range of levels without an increment.

ERROR MESSAGES:

[LV +1] More than NEXP values found.
[LV -1] Level cannot be decoded.
[LV -2] The vertical coordinate for ... must be PRES.
[LV -3] Invalid vertical coordinate.
[LV -4] Invalid input for LEVEL.
[LV -5] Range with increment cannot include SFC (0).

LEVEL (LV) LIBRARY

LV Library Calls

LV_CCRD	(ivcord, / vcoord, iret)
LV_CORD	(vcoord, / vparm, ivert, iret)
LV_DECD	(clevel, / rlev, iret)
LV_DUPL	(/ nlev, rlevel, / iret)
LV_GRNG	(start, stop, inc, nexp, / rlevel, / nlev, iret)
LV_INPT	(level, nexp, vcoord, / nlev, rlevel, levtyp, vparm, ivert, iret)
LV_MANL	(nexp, / nlev, rlevel, iret)
LV_SORT	(ivert, / nlev, rlevel, / iret)
LV_VASL	(nexp, / nlev, rlevel, iret)

13.1 LV_CCRD - GET VERTICAL COORD NAME

This subroutine translates a numeric value for IVCORD into its character value in VCOORD.

LV_CCRD (IVCORD, VCOORD, IRET)

Input parameters:

IVCORD	INTEGER	Numeric vertical coordinate

Output parameters:

VCOORD	CHAR*	Vertical coordinate
IRET	INTEGER	Return code 0 = normal return -3 = invalid coordinate

13.2 LV_CORD - GET VERTICAL COORDINATE

This subroutine converts the input for VCOORD to upper-case and translates it to a numeric value.

LV_CORD (VCOORD, VPARM, IVERT, IRET)

Input parameters:

VCOORD	CHAR*	Vertical coordinate input

Output parameters:

VPARM	CHAR*	Upper-case coordinate
IVERT	INTEGER	Numeric vertical coordinate 0 = NONE 1 = PRES 2 = THTA 3 = HGHT
IRET	INTEGER	Return code 0 = normal return -3 = invalid coordinate

13.3 LV_DECD - DECODE SINGLE LEVEL

This subroutine decodes a single level. CLEVEL must be a number, SFC, or TOP. SFC and TOP will be transformed into 0 and -1, respectively.

LV_DECD (CLEVEL, RLEV, IRET)

Input parameters:

CLEVEL	CHAR*	Input character level

Output parameters:

RLEV	REAL	Level value
IRET	INTEGER	Return code 0 = normal return -1 = decode error

13.4 LV_DUPL - ELIMINATE DUPLICATE LEVELS

This subroutine eliminates duplicate levels from a list of levels. The variables NLEV and RLEVEL are updated.

LV_DUPL (NLEV, RLEVEL, IRET)

Input and output parameters:

NLEV	INTEGER	Number of levels
RLEVEL (NLEV)	REAL	Levels

Output parameters:

IRET	INTEGER	Return code 0 = normal return

13.5 LV_GRNG - GET LEVELS FROM RANGE

This subroutine finds the levels in a range with an increment. START, STOP and INC are decoded as integers and the levels are computed.

LV_GRNG (START, STOP, INC, NEXP, RLEVEL, NLEV, IRET)

Input parameters:

START	CHAR*	Start of range
STOP	CHAR*	End of range
INC	CHAR*	Increment
NEXP	INTEGER	Maximum number of levels

Output parameters:

RLEVEL (NLEV)	REAL	Levels in range
NLEV	INTEGER	Number of levels
IRET	INTEGER	Return code 1 = more than NEXP values 0 = normal return -1 = input cannot be decoded -5 = 0 invalid in range w inc

13.6 LV_INPT - DECODE INPUT FOR LEVEL AND VCOORD

This subroutine converts the user input for LEVELS and VCOORD into a list of levels. The input for LEVELS may be a list separated by semicolons. The following items may be included in the list:

a single level;
MAN for the mandatory levels below 100 mb;
VAS for the standard VAS levels;
a range of levels with an increment.

The following items are also valid, provided they are not part of a list:

ALL for all levels;
a range of levels without an increment.

If a range without an increment is entered, the limits will be returned in RLEVEL and LEVTYP will be set to 2.

If MAN or VAS is input, the input vertical coordinate must be PRES. The names SFC and TOP may be used. They will be translated into 0 and -1, respectively.

```
LV_INPT ( LEVEL, NEXP, VCOORD, NLEV, RLEVEL, LEVTYP, VPARM,
          IVERT, IRET )
```

Input parameters:

LEVEL	CHAR*	Input for LEVEL
NEXP	INTEGER	Maximum number of levels
VCOORD	CHAR*	Input for VCOORD

Output parameters:

NLEV	INTEGER	Number of levels
RLEVEL (NLEV)	REAL	Levels or range
LEVTYP	INTEGER	Level type 0 = no levels input 1 = list of levels 2 = range of levels
VPARM	CHAR*	Vertical coordinate
IVERT	INTEGER	Numerical vertical coord 0 = NONE 1 = PRES 2 = THTA 3 = HGHT
IRET	INTEGER	Return code 1 = too many levels 0 = normal return -2 = MAN, VAS need PRES cord

-3 = invalid VCOORD
-4 = invalid input for LEVEL
-5 = range w inc can't have 0

13.7 LV_MANL - RETURN MANDATORY LEVELS

This subroutine returns the mandatory levels below 100 mb.

LV_MANL (NEXP, NLEV, RLEVEL, IRET)

Input parameters:

NEXP	INTEGER	Maximum number of levels

Output parameters:

NLEV	INTEGER	Number of levels
RLEVEL (NLEV)	REAL	Levels
IRET	INTEGER	Return code 1 = more than NEXP values 0 = normal return

13.8 LV_SORT - SORT LEVELS

This subroutine sorts levels from the surface to the top of the atmosphere. They are sorted in descending order if IVERT = 1; otherwise the levels are in ascending order. In either case, the surface level (RLEVEL = 0) is first and the top level (RLEVEL = -1) is last. Duplicate levels are eliminated.

LV_SORT (IVERT, NLEV, RLEVEL, IRET)

Input parameters:

IVERT	INTEGER	Numeric vertical coordinate 0 = NONE 1 = PRES 2 = THTA 3 = HGHT

Input and output parameters:

NLEV	INTEGER	Number of levels
RLEVEL (NLEV)	REAL	Vertical levels

Output parameters:

IRET	INTEGER	Return code 0 = normal return

13.9 LV_VASL - RETURN VAS LEVELS

This subroutine returns the standard VAS levels of 1000, 920, 850, 700, 600, 500, 400, 350, 300, 250, 200, 175, 150, 125 and 100 mb.

LV_VASL (NEXP, NLEV, RLEVEL, IRET)

Input parameters:

NEXP	INTEGER	Maximum number of levels

Output parameters:

NLEV	INTEGER	Number of levels
RLEVEL (NLEV)	REAL	Levels
IRET	INTEGER	Return code 1 = more than NEXP values 0 = normal return

CHAPTER 14

UPPER-AIR MERGE (MR) LIBRARY

Upper-Air Merge (MR) Library

The upper-air merge library merges mandatory and significant level upper-air reports into a single station sounding. This library is called by the SN subroutines to merge data which is stored as separate parts in an SN file.

The main subroutine used to merge data is MR_UADT. This subroutine takes reports for mandatory, significant temperature and significant wind data, both below and above 100 mb, and creates a sounding where all the input levels are present with data interpolated to these levels, if necessary. The order of the parameters in the input reports is critical to the proper execution of this subroutine.

The data is merged using the following sequence.

1. The surface data is found by checking the TTAA, TTBB and PPBB reports.
2. The below- and above-100-mb mandatory (TTAA, TTCC) data reports are combined.
3. The significant temperature (TTBB, TTDD) reports are merged with the speed, direction and height set to missing.
4. The height at all levels is recomputed using one of the three methods described below.
5. The significant wind (PPBB, PPDD) reports are merged using the heights from the reports and the pressure is computed by interpolating the LOG (pressure) linearly with height. If the significant wind data was reported on pressure surfaces, it is merged using pressure, and the heights are then computed.
6. Missing values of speed, direction, temperature and dewpoint are computed by interpolating linearly with respect to LOG (pressure).

One of three methods of height interpolation can be specified in IZTYPE. These are:

IZTYPE = 1 The height is computed by interpolating linearly with respect to LOG (pressure). If heights cannot be interpolated at the top of the sounding, they are computed using method 2.

IZTYPE = 2 The heights are replaced with the moist hydrostatic height computed using the temperature. Heights which were reported at mandatory levels are replaced by the computed heights.

IZTYPE = 3 The heights reported at mandatory levels are retained. Heights at levels between mandatory levels are computed as moist hydrostatic heights and scaled to fit the mandatory heights. Above the top mandatory report, heights are computed using method 2.

The SN subroutine which merges data transparently sets IZTYPE to 3.

MR Library Calls

MR_UADT (datman, nman, datsgt, nsgt, datsgw, nsgw, datamn, namn, datast, nast, datasw, nasw, selv, iztype, / stndat, nlev, idtype, iret)

14.1 MR_UADT - MERGE UPPER-AIR DATA

This subroutine merges mandatory and significant level data.

```
MR_UADT  ( DATMAN, NMAN, DATSGT, NSGT, DATSGW, NSGW, DATAMN, NAMN,
           DATAST, NAST, DATASW, NASW, SELV, IZTYPE, STNDAT, NLEV,
           IDTYPE, IRET )
```

Input parameters:

DATMAN (6,NMAN)	REAL	Mandatory data below 100 mb
NMAN	INTEGER	Number of levels
DATSGT (3,NSGT)	REAL	Sig temp data below 100 mb
NSGT	INTEGER	Number of levels
DATSGW (3,NSGW)	REAL	Sig wind data below 100 mb
NSGW	INTEGER	Number of levels
DATAMN (6,NAMN)	REAL	Mandatory data above 100 mb
NAMN	INTEGER	Number of levels
DATAST (3,NAST)	REAL	Sig temp data above 100 mb
NAST	INTEGER	Number of levels
DATASW (3,NASW)	REAL	Sig wind data above 100 mb
NASW	INTEGER	Number of levels
SELV	REAL	Surface elevation
IZTYPE	INTEGER	Type of height interpolation 1 = int wrt log p 2 = moist hydrostatic comp 3 = scaled moist hydro comp

Output parameters:

STNDAT (6,NLEV)	REAL	Station data
NLEV	INTEGER	Number of levels
IDTYPE (NLEV)	INTEGER	Data type flags 1 = mandatory 2 = sig temperature 3 = sig wind
IRET	INTEGER	Return code 0 = normal return

CHAPTER 15

NON-TAE (NT) LIBRARY

NT_DFLT	Get defaults
NT_DFTS	Get system defaults
NT_DYNM	Dynamic tutor
NT_EXIT	Non-TAE exit
NT_HELP	Non-TAE help
NT_IDNT	Program identification
NT_INIT	Initialize non-TAE
NT_LIST	List variables
NT_REST	Restore parameter file
NT_RQST	Parameters requested
NT_SAVE	Save parameter file
NT_STR	Get variable value
NT_STRP	Read .PDF variables
NT_SVAR	Define variable
NT_ULOC	Update local variable

Non-TAE (NT) Library

The non-TAE library contains subroutines to replace the TAE calls. This library initializes parameter values from a file called NOTAE.GLB. If the file is not present, it will be created and defaults from GEMGLB.PDF will be used. NT_EXIT updates the values of the parameters in the file. Since all parameters are saved in the file, there is no distinction between global and local values.

When using the non-TAE interface, a dynamic tutor is available with the following commands:

LIST	list values for program parameters
LIST PARM	list value for parameter PARM
PARM=VALUE	changes parameter value
HELP	gives help on program
HELP PARM	write the parameter help file
RUN	execute the program
EXIT	exit the program
SAVE FILE	saves current parameters in FILE.NTS
RESTORE FILE	restores parameters from FILE.NTS

These subroutines are called by the IP library and should not be called directly by applications programs.

ERROR MESSAGES:

[NT -1]	NOTAE.GLB cannot be opened for write access.
[NT -2]	The global parameter file GEMGLB cannot be opened.
[NT -3]	There are too many parameters.
[NT -4]	Error writing to NOTAE.GLB.
[NT -5]	... is an unrecognized command.
[NT -6]	There is no help file for ....
[NT -7]	... is an unrecognized parameter.
[NT -8]	... is an ambiguous parameter abbreviation.
[NT -9]	The save file ... is invalid.
[NT -10]	The parameter file for ... cannot be opened.

NT Library Calls

NT_DFLT	(/ iret)
NT_DFTS	(/ iret)
NT_DYNM	(/ done, iret)
NT_EXIT	(/ iret)
NT_HELP	(pname, / iret)
NT_IDNT	(progrm, / iret)
NT_INIT	(/ iret)
NT_LIST	(pname, list, / iret)
NT_REST	(file, / iret)
NT_RQST	(/ iret)
NT_SAVE	(file, / iret)
NT_STR	(pname, / parm, iret)
NT_STRP	(progrm, / iret)
NT_SVAR	(input, / iret)
NT_ULOC	(pname, parm, / iret)

15.1 NT_DFLT - GET DEFAULTS

This subroutine gets default values for the TAE parameters. If the local NOTAE.GLB file cannot be read, GEMGLB.PDF is read.

NT_DFLT (IRET)

Output parameters:

IRET	INTEGER	Return code
		0 = normal return
		-2 = unable to open GEMGLB
		-3 = too many variables

15.2 NT_DFTS - GET SYSTEM DEFAULTS

This subroutine reads the global parameters and their default values from the GEMPAK system file GEMGLB.PDF.

NT_DFTS (IRET)

Output parameters:

IRET	INTEGER	Return code 0 = normal return -2 = unable to open GEMGLB

15.3 NT_DYNM - DYNAMIC TUTOR

This subroutine replaces the dynamic tutor for the non-TAE user. Error messages which are encountered in processing user input will be written to the user's terminal.

NT_DYNM (DONE, IRET)

Output parameters:

DONE	LOGICAL	Program exit flag
IRET	INTEGER	Return code 0 = normal return

15.4 NT_EXIT - NON-TAE EXIT

This subroutine must be called to exit from the non-TAE. Current variable values are written to the file containing global values.

NT_EXIT (IRET)

Output parameters:

IRET	INTEGER	Return code 0 = normal return -1 = unable to open NOTAE.GLB -4 = error writing to file

15.5 NT_HELP - NON-TAE HELP

This subroutine writes a help file for a variable. If PNAME is blank, help for the program will be written.

NT_HELP (PNAME, IRET)

Input parameters:

PNAME	CHAR*	Variable

Output parameters:

IRET	INTEGER	Return code 0 = normal return -6 = no help available

15.6 NT_IDNT - PROGRAM IDENTIFICATION

This subroutine saves the name of the program being executed. The parameters used in the program are read from the PDF file. If $RESPOND is set, a dynamic tutor is entered.

NT_IDNT (PROGRM, IRET)

Input parameters:		
PROGRM	CHAR*	Program name
Output parameters:		
IRET	INTEGER	Return code 0 = normal return -10 = pdf cannot be opened

15.7 NT_INIT - INITIALIZE NON-TAE

This subroutine does the initialization for the non-TAE mode. The file NOTAE.GLB is read to obtain global variable values. An error opening the file with write access will prevent global values from being saved.

NT_INIT (IRET)

Output parameters:

IRET	INTEGER	Return code 0 = normal return -1 = too many variables

15.8 NT_LIST - LIST VARIABLES

This subroutine lists the current value for the variable. If PNAME is blank, all the variables for the current program will be listed.

NT_LIST (PNAME, IRET)

Input parameters:

PNAME	CHAR*	Variable name
LIST	LOGICAL	List flag

Output parameters:

IRET	INTEGER	Return code 0 = normal return

15.9 NT_REST - RESTORE PARAMETER FILE

This subroutine restores a parameter file for the non-TAE.

NT_REST (FILE, IRET)

Input parameters:

FILE	CHAR*	Input file name

Output parameters:

IRET	INTEGER	Return code 0 = normal return -9 = invalid file name

15.10 NT_RQST - PARAMETERS REQUESTED

This subroutine lists the variables used in the program.

NT_RQST (IRET)

Output parameters:

IRET	INTEGER	Return code 0 = normal return

15.11 NT_SAVE - SAVE PARAMETER FILE

This subroutine saves a parameter file for the non-TAE.

NT_SAVE (FILE, IRET)

Input parameters:

FILE	CHAR*	Input file name

Output parameters:

IRET	INTEGER	Return code 0 = normal return -9 = invalid file name

15.12 NT_STR - GET VARIABLE VALUE

This subroutine receives a string variable from the non-TAE. If the parameter does not have a value, PARM is set to blanks. If no more variables can be added to the global table, an error message is written and IRET is set to -3.

NT_STR (PNAME, PARM, IRET)

Input parameters:

PNAME	CHAR*	Variable name

Output parameters:

PARM	CHAR*	Variable value
IRET	INTEGER	Return code 0 = normal return -1 = too many variables -3 = too many parameters

15.13 NT_STRP - READ .PDF VARIABLES

This subroutine receives the name of the program being executed and reads the program's .PDF file in GEMEXE to make a list of the variables to be used. If a variable from the PDF file is also in the NOTAE.GLB, then a flag is set showing it is being used. Otherwise, the variable is added to the global list.

NT_STRP (PROGRM, IRET)

Input parameters:

PROGRM	CHAR*	Program name

Output parameters:

IRET	INTEGER	Return code 0 = normal return -10 = unable to open PDF file

15.14 NT_SVAR - DEFINE VARIABLE

This subroutine sets the value of a variable in a dynamic tutor. It assumes that input is in the form VAR=VALUE. The variable may be a unique abbreviation of a current program variable. If the variable is any other global, the full name must be input.

NT_SVAR (INPUT, IRET)

Input parameters:

INPUT	CHAR*	Input string

Output parameters:

IRET	INTEGER	Return code 0 = normal return

15.15 NT_ULOC - UPDATE LOCAL VARIABLE

This subroutine saves the value of a variable.

NT_ULOC (PNAME, PARM, IRET)

Input parameters:

PNAME	CHAR*	Variable name
PARM	CHAR*	Variable value

Output parameters:

IRET	INTEGER	Return code 0 = normal return

CHAPTER 16

OBJECTIVE ANALYSIS (OA) LIBRARY

OA_BARN	Perform Barnes analysis
OA_BOXC	Compute station row/col
OA_GFIL	Open grid file
OA_GUES	Put guess on analysis grid
OA_IGRD	Initialize grid to zero
OA_LTLN	Compute lat/lon for grid
OA_NAVG	Compare grid navigations
OA_SINT	Interpolate grid to stations
OA_WFSR	Compute weighting function
OA_WGRD	Write grids to file
OA_WRMS	Write RMS errors

Objective Analysis (OA) Library Summary

The objective analysis library performs general objective analysis functions.

Three areas are defined in the objective analysis programs:

GRID	Grid area	Computed grid area
EXTEND	Extend grid	Grid area extended for first pass
DATA	Data area	Data subset area.

These areas are specified by the lower left and upper right corners. If the grid projection is not a latitude/longitude (CED) projection, the range for the grid or extend area may not be identical with the range in GRID or EXTEND.

Information about the grid area is obtained from the grid navigation block which is stored in the grid file. The data area and extend area are stored in the analysis block. Two types of analyses blocks are available. The contents of these blocks are:

WORD	CONTENTS for type 1	CONTENTS for type 2
1	type = 1.0	type = 2.0
2	DELTAN	DELTAN
3	DELTAX	IEXTND (1)
4	DELTAY	IEXTND (2)
5	GAMMA (not used)	IEXTND (3)
6	GBNDS (1)	IEXTND (4)
7	GBNDS (2)	GBNDS (1)
8	GBNDS (3)	GBNDS (2)
9	GBNDS (4)	GBNDS (3)
10	EBNDS (1)	GBNDS (4)
11-13	EBNDS (2-4)	EBNDS (1-3)
14-17	DBNDS (1-4)	EBNDS (4), DBNDS (1-3)
18		DBNDS (4).

These variables have the following meanings:

DELTAN	-	station spacing (degrees)
DELTAX	-	spacing in x direction (degrees)
DELTAY	-	spacing in y direction (degrees)
IEXTND	-	number of points to extend grid
GBNDS	-	grid area corners
EBNDS	-	extend grid area corners
DBNDS	-	data subset corners.

Note that analysis type 1 assumes that the grid projection is CED.

ERROR MESSAGES

[OA -1]	Grid file ... could not be opened.
[OA -2]	Invalid grid projection area.
[OA -3]	Guess file ... could not be opened.
[OA -4]	Analysis and guess grids have different navigations.
[OA -5]	Guess grid could not be found.
[OA -6]	Guess grid does not align with unextended analysis grid.

OA Library Calls

OA_BARN	(ngrid, weight, srad, kexy, nstn, data, slat, slon, gelat, gelon, coslsq, / isn, grid, iret)
OA_BOXC	(slat, slon, nstn, iextnd, / srow, scol, iret)
OA_GFIL	(gdfile, / gdcur, guess, gscur, / igdfln, igdgs, gsflag, gsdttm, deltan, gltln, kex, key, iextnd, iret)
OA_GUES	(gg, ix, iy, iextnd, / ag, ing, ngrid, kex, key, iret)
OA_IGRD	(ngrid, kex, key, kexkey, iextnd, gadttm, iglvl, igvcr, gparm, gsflag, gsdttm, / grid, iret)
OA_LTLN	(kex, key, iextnd, / gelat, gelon, coslsq, iret)
OA_NAVG	(rnvblk, gsnvbk, navsz, / gsflag, iret)
OA_SINT	(ngrid, nstn, data, srow, scol, kex, key, grid, iextnd, / sdint, rms, isn, iret)
OA_WFSR	(deltan, search, / weight, srad, iret)
OA_WGRD	(igdfln, gdattm, ngrid, parm, level, ivcord, grid, kex, key, iextnd, / iret)
OA_WRMS	(ipass, parms, nparms, levels, nlev, rms, isn, / iret)

16.1 OA_BARN - PERFORM BARNES ANALYSIS

This subroutine performs a single pass of a Barnes analysis. The weighting function used is

[EXP (DIST ** 2 / WEIGHT)]

where DIST is the distance from the station to the grid point. Locations and distances in this subroutine are now defined in latitude/longitude. Distance between a grid point and a station is computed as

```
DIST ** 2 = [ ( lat (grid) - lat (stn) ) ** 2 +
            ( ( lon (grid) - lon (stn) ) ** 2 ) * coslsq (grid) ) ]
```

ISN is the number of stations used for each grid computed. Only data within the distance [SQRT (SRAD)] of a grid point is included in the analysis.

```
OA_BARN ( NGRID, WEIGHT, SRAD, KEXY, NSTN, DATA, SLAT, SLON,
          GELAT, GELON, COSLSQ, ISN, GRID, IRET )
```

Input parameters:

NGRID	INTEGER	Number of grids
WEIGHT	REAL	Weighting factor
SRAD	REAL	Search radius in grid coords
KEXY	INTEGER	# of points in extend grid
NSTN	INTEGER	Number of stations
DATA (NGRID,NSTN)	REAL	Station data
SLAT (NSTN)	REAL	Station latitudes
SLON (NSTN)	REAL	Station longitudes
GELAT (KEXY)	REAL	Grid point latitudes
GELON (KEXY)	REAL	Grid point longitudes
COSLSQ (KEXY)	REAL	COS (lat) squared at grid pts

Output parameters:

ISN (NGRID)	INTEGER	# stations used for grid
GRID (NGRID,KEXY)	REAL	Grid data
IRET	INTEGER	Return code 0 = normal return

16.2 OA_BOXC - COMPUTE STATION ROW/COL

This subroutine translates station latitude and longitude into the row and column numbers in the extend grid. The first two values in IEXTND are used to translate grid coordinates to the extend grid coordinates.

OA_BOXC (SLAT, SLON, NSTN, IEXTND, SROW, SCOL, IRET)

Input parameters:

SLAT (NSTN)	REAL	Station latitudes
SLON (NSTN)	REAL	Station longitudes
NSTN	INTEGER	Number of stations
IEXTND (4)	INTEGER	Extend grid points

Output parameters:

SROW (NSTN)	REAL	Rows in extend grid
SCOL (NSTN)	REAL	Columns in extend grid
IRET	INTEGER	Return code 0 = normal return -2 = invalid grid projection

16.3 OA_GFIL - OPEN GRID FILE

This subroutine opens a grid file and returns the information needed to perform a Barnes objective analysis. The grid area is obtained from the navigation block. The extend area and the station spacing are obtained from the analysis block. The first-guess grid file is also opened and a check is done to see that the guess file and the analysis file have the same navigation. If a guess file exists, DG_INIT is called.

```
OA_GFIL ( GDFILE, GUESS, GDCUR, GSCUR, IGDFLN, IGDGS, GSFLAG,
          GSDTTM, DELTAN, GLTLN, KEX, KEY, IEXTND, IRET )
```

Input parameters:

GDFILE	CHAR*	Grid file
GUESS	CHAR*	First guess grid file/time

Input and output parameters:

GDCUR	CHAR*	Current grid file
GSCUR	CHAR*	Current guess grid file

Output parameters:

IGDFLN	INTEGER	Grid file number
IGDGS	INTEGER	Guess file number
GSFLAG	LOGICAL	Guess file existence flag
GSDTTM	CHAR*	Guess GEMPAK time
DELTAN	REAL	Station spacing
GLTLN (4)	REAL	Grid area
KEX	INTEGER	Number of x points in ELTLN
KEY	INTEGER	Number of y points in ELTLN
IEXTND (4)	INTEGER	Grid extension size
IRET	INTEGER	Return code 0 = normal return -1 = grid file open error -3 = guess file open error -4 = navigation discrepancy

16.4 OA_GUES - PUT GUESS ON ANALYSIS GRID

This subroutine puts the guess grid into the analysis grid.

OA_GUES (GG, IX, IY, IEXTND, AG, ING, NGRID, KEX, KEY, IRET)

Input parameters:

GG (IX, IY)	REAL	Guess grid
IX	INTEGER	Guess grid x dimension
IY	INTEGER	Guess grid y dimension
IEXTND (4)	INTEGER	Grid extension specification
ING	INTEGER	Position to load analysis grid
NGRID	INTEGER	Total number of grid positions
KEX	INTEGER	Analysis grid x dimension
KEY	INTEGER	Analysis grid y dimension

Output parameters:

AG (NGRID, KEX, KEY)	REAL	Analysis grid (incl extension)
IRET	INTEGER	Return code 0 = normal return -6 = grid alignment error

16.5 OA_IGRD - INITIALIZE GRID TO ZERO

This subroutine initializes grid data to either zero or first-guess grid values.

OA_IGRD (NGRID, KEX, KEY, KEXKEY, IEXTND, GADTTM, IGLVL, IGVCR, GPARM, GSFLAG, GSDTTM, GRID, IRET)

Input parameters:

NGRID	INTEGER	Number of grids
KEX	INTEGER	X dimension of extend grid
KEY	INTEGER	Y dimension of extend grid
KEXKEY	INTEGER	KEX * KEY
IEXTND (4)	INTEGER	Extend region specification
GADTTM	CHAR*	Analysis time
IGLVL (NGRID)	INTEGER	Levels of grids
IGVCR (NGRID)	INTEGER	Vertical coordinates 0 = NONE 1 = PRES 2 = THTA 3 = HGHT
GPARM (NGRID)	CHAR*	Parameters
GSFLAG	LOGICAL	Flag for first guess
GSDTTM	CHAR*	Guess time

Output parameters:

GRID (NGRID, KEXKEY)	REAL	Grid data
IRET	INTEGER	Return code 0 = normal return -5 = guess does not exist

16.6 OA_LTLN - COMPUTE LAT/LON FOR GRID

This subroutine computes the latitude and longitude at each grid point in the extend grid area. The grid coordinate system must be defined in GEMPLT before this subroutine is called.

OA_LTLN (KEX, KEY, IEXTND, GELAT, GELON, COSLSQ, IRET)

Input parameters:

KEX	INTEGER	# of x points in extend grid
KEY	INTEGER	# of y points in extend grid
IEXTND (4)	INTEGER	# of points to extend grid

Output parameters:

GELAT (KEX,KEY)	REAL	Latitudes in degrees
GELON (KEX,KEY)	REAL	Longitudes in degrees
COSLSQ(KEX,KEY)	REAL	Cosine of latitude squared
IRET	INTEGER	Return code 0 = normal return -2 = grid projection error

16.7 OA_NAVG - COMPARE GRID NAVIGATIONS

This subroutine checks the navigation block of the analysis grid against that of the guess grid.

OA_IGRD (RNVBLK, GSNVBK, NAVSZ, GSFLAG, IRET)

Input parameters:

RNVBLK (NAVSZ)	REAL	Analysis grid navigation block
GSNVBK (NAVSZ)	REAL	Guess grid navigation block
NAVSZ	INTEGER	Navigation length

Output parameters:

GSFLAG	LOGICAL	Check flag
IRET	INTEGER	Return code 0 = normal return -4 = no match

16.8 OA_SINT - INTERPOLATE GRID TO STATIONS

This subroutine interpolates data from a grid back to the stations using a bilinear interpolation, and computes the difference between the original data and the interpolated values. Data are interpolated to all stations in the extend area, but only stations within the grid area are used to compute the RMS values. ISN is the number of stations used to compute the RMS value.

```
OA_SINT ( NGRID, NSTN, DATA, SROW, SCOL, KEX, KEY, GRID, IEXTND,
          SDINT, RMS, ISN, IRET )
```

Input parameters:

NGRID	INTEGER	Number of grids
NSTN	INTEGER	Number of stations
DATA (NGRID,NSTN)	REAL	Station data
SROW (NSTN)	REAL	Station rows
SCOL (NSTN)	REAL	Station columns
KEX	INTEGER	X points in extend grid
KEY	INTEGER	Y points in extend grid
GRID (NGRID,KEX,KEY)	REAL	Grid data
IEXTND (4)	INTEGER	Grid area extensions

Output parameters:

SDINT (NGRID,NSTN)	REAL	Station differences
RMS (NGRID)	REAL	RMS values
ISN (NGRID)	INTEGER	# of stations in grid area
IRET	INTEGER	Return code 0 = normal return

16.9 OA_WFSR - COMPUTE WEIGHTING FUNCTION

This subroutine computes the weighting factor and search radius to be used in the Barnes analysis. The weighting factor is computed using the formula:

WEIGHT = [5.051457 * (DELTAN * 2. / PI) ** 2]

The search radius, SRAD, is computed as SEARCH * WEIGHT. This limits the search area to stations whose weights will be larger than [EXP (-SEARCH)] . If SEARCH is non-positive, a value of 20 will be used. Both the weighting factor and search radius should be multiplied by GAMMA for the second pass analysis.

OA_WFSR (DELTAN, SEARCH, WEIGHT, SRAD, IRET)

Input parameters:

DELTAN	REAL	Station spacing
SEARCH	REAL	User input search condition

Output parameters:

WEIGHT	REAL	Weighting factor
SRAD	REAL	Search radius
IRET	INTEGER	Return code 0 = normal return

16.10 OA_WGRD - WRITE GRIDS TO FILE

This subroutine writes grids computed in an objective analysis to the grid file. Although the data in GRID have been computed for the extend area, only the data in the grid area are written to the file.

```
OA_WGRD ( IGDFLN, GDATTM, NGRID, PARM, LEVEL, IVCORD, GRID, KEX,
          KEY, IEXTND, IRET )
```

Input parameters:

IGDFLN	INTEGER	Grid file number
GDATTM	CHAR*	Date/time for grids
NGRID	INTEGER	Number of grids
PARM (NGRID)	CHAR*	Parameters
LEVEL (NGRID)	INTEGER	Levels
IVCORD (NGRID)	INTEGER	Vertical coordinate
GRID (NGRID,KEX,KEY)	REAL	Grid data
KEX	INTEGER	X points in extend area
KEY	INTEGER	Y points in extend area
IEXTND (4)	INTEGER	Grid extension

Output parameters:

IRET	INTEGER	Return code 0 = normal return

16.11 OA_WRMS - WRITE RMS ERRORS

This subroutine writes RMS values from an objective analysis.

OA_WRMS (IPASS, PARMS, NPARMS, LEVELS, NLEV, RMS, ISN, IRET)

Input parameters:

IPASS	INTEGER	Pass number
PARMS (NPARMS)	CHAR*	Parameters
NPARMS	INTEGER	Number of parameters
LEVELS (NLEV)	INTEGER	Levels
NLEV	INTEGER	Number of levels
RMS (NPARMS,NLEV)	REAL	RMS values
ISN (NPARMS,NLEV)	INTEGER	Number of stations reporting

Output parameters:

IRET	INTEGER	Return code 0 = normal return

CHAPTER 17

PARAMETER CONVERSION (PC) LIBRARY

PC_CMLV	Compute station and level parms
PC_CMST	Compute station data
PC_CMVR	Compute data for vertical level
PC_CMVS	Compute data for level number
PC_DFLS	Define station and level parms
PC_DFLV	Define computed level parameters
PC_DFST	Define computed station parameters
PC_FLVL	Find levels in a coordinate system
PC_INIT	Initialize PC subroutines
PC_INTP	Interpolate between two levels
PC_SLCD	Define level conditions
PC_SSCD	Define station conditions
PC_SSTN	Define station information
PC_STIM	Set date/time

Parameter Conversion (PC) Library Summary

The Parameter Conversion Library is used to compute the desired meteorological parameters for both upper-air and surface programs.

In order to use the package the programmer must call PC_INIT first. This subroutine provides information about the data set to the conversion package. In particular, it specifies the names of the parameters which are included in the data set. If PC_INIT is called a second time, all calls to define output parameters must be repeated.

PC_STIM defines the current data set time to GEMPAK. It is not used currently, but parameters which encode the time may be defined in the future.

There are two types of parameters which can be computed by the PC subroutines. LEVEL parameters include parameters computed at a specific level of the atmosphere as well as layer parameters which are computed for a layer specified by two levels. Level parameters at a specific level include TMPC and MIXR. Layer parameters, such as RICH and BVFQ, use the two significant levels bounding the input level. STATION parameters, such as SELV and stability indices, have a single value associated with the station.

Conditions may now be defined for each type of parameter. These conditions are defined in the subroutines listed below. The conditional functions >, < and = will return data only if the condition is met. For example, TMPC > 0 will only return reports at levels where the temperature is greater than or equal to 0. The conditional functions +, -, *, and / will perform the required function on the specified parameters. For example, if TMPC = 12.34, TMPC * 10 will return 123.4. Finally, parameters requiring user input use the symbols !, % and $. By convention, ! precedes a layer depth, % precedes the numerical value of the vertical coordinate and $ specifies a storm direction.

The PC library contains subroutines to define parameters to be computed and corresponding subroutines to perform the computations. The following chart lists these subroutines and the types of parameters for which they are designed.

The STATION & LEVEL subroutines should be used whenever the parameters are to be returned in a single array and the distinction between level and station parameters is important. Programs such as SNMAP and SFLIST call these subroutines.

PARAMETER CONVERSION (PC) LIBRARY

TYPE	DEFINE	COMPUTE	CONDITIONS
STATION & LEVEL	PC_DFLS	PC_CMVS	PC_SLCD
STATION	PC_DFST	PC_CMST	PC_SSCD
LEVEL	PC_DFLV	PC_CMLV PC_CMVR	PC_SLCD

All of the DEFINE subroutines will initialize the compute flags, CMPFLG. Any parameter which cannot be identified as computable by the DEFINE subroutine called will be set to FALSE. When the COMPUTE subroutines are called, any non-computable parameter will be returned with the missing data value. This is a change from earlier versions of the PC subroutines. Note that, if PC_DFST and PC_DFLV are to be called separately, PC_DFLV must be called first.

In all cases, PC_SSTN must be called to save the station information before any COMPUTE subroutine is called.

The routines PC_FLVL and PC_INTP are also available for general use.

ERROR MESSAGES:

[PC +21] There are too many parameters in parameter type table.
[PC +20] There are too many functions in table for internal buffer.
[PC -1] NPARM is invalid.
[PC -2] IVERT is invalid.
[PC -3] NLEV is invalid.
[PC -4] PC_INIT must be called.
[PC -5] Invalid number of output parameters.
[PC -6] PC_SSTN must be called first.
[PC -7] Output parameters have not been defined.
[PC -8] Requested level number is out of range.
[PC -9] Surface data requested but not in dataset.
[PC -10] Only surface data can be computed when IVERT = 0.
[PC -11] No valid data at station.
[PC -13] Data cannot be computed for output vertical coordinate.
[PC -19] No interpolations can be done since PRES is not computable.
[PC -20] Function table cannot be opened.
[PC -21] Error opening parameter type table.
[PC -22] Invalid index in PC_COMP.
[PC -23] Data to compute isentropic data is missing.
[PC -24] Interpolation PRES is not between two input levels.
[PC -26] Isentropic data is not computable.
[PC -27] Computation of THTA does not converge.
[PC -28] Input PRES of 0 is invalid.
[PC -30] Invalid index.

PC Library Calls

```
PC_CMLV   ( levnum, datain, / outdat, chrdat, iret )
PC_CMST   ( datain, / outdat, chrdat, iret )
PC_CMVR   ( vlev, ivcord, datain, / outdat, chrdat, iret )
PC_CMVS   ( vlev, ivcord, datain, / outdat, chrdat, iret )
PC_DFLS   ( noutpm, outprm, chrflg, cmpflg, levflg, ncomp, iret )
PC_DFLV   ( noutpm, outprm, chrflg, cmpflg, ncomp, iret )
PC_DFST   ( noutpm, outprm, chrflg, cmpflg, ncomp, iret )
PC_FLVL   ( vlev, ivcord, datain, / rlev, level1, level2, levtyp,
            iret )
PC_INIT   ( ivert, nparm, parms, / iret )
PC_INTP   ( vlev, adata, bdata, nparms, intflg, angflg, / outdat,
            iret )
PC_SLCD   ( noutpm, condtn, / iret )
PC_SSCD   ( noutpm, condtn, / iret )
PC_SSTN   ( stid, isnum, slat, slon, selv, ihhmm, nlev, / iret )
PC_STIM   ( ihhmm, / iret )
```

17.1 PC_CMLV - COMPUTE STATION AND LEVEL PARMS

This subroutine computes level parameters for a particular data set level specified by the level number. Only level data are computed. The output parameters must be defined by a call to PC_DFLV before this subroutine is called.

PC_CMLV (LEVNUM, DATAIN, OUTDAT, CHRDAT, IRET)

```
Input parameters:
   LEVNUM              INTEGER          Level number
   DATAIN              REAL             Station data
     (NPARM,NLEV)

Output parameters:
   OUTDAT (NOUTPM)     REAL             Computed real data
   CHRDAT (NOUTPM)     CHAR*            Computed character data
   IRET                INTEGER          Return code
                                          0 = normal return
                                         -4 = PC_INIT not called
                                         -6 = PC_SSTN not called
                                         -7 = output parms not set
                                         -8 = invalid level number
```

17.2 PC_CMST - COMPUTE STATION DATA

This subroutine computes station parameters. PC_DFST must be called to define the output parameters before this subroutine is called.

PC_CMST (DATAIN, OUTDAT, CHRDAT, IRET)

Input parameters:

DATAIN (NPARM,NLEV)	REAL	Station data

Output parameters:

OUTDAT (NSPRM)	REAL	Computed real data
CHRDAT (NSPRM)	CHAR*	Computed character data
IRET	INTEGER	Return code 0 = normal return -4 = PC_INIT not called -6 = PC_SSTN not called -16 = no stn parameters set

17.3 PC_CMVR - COMPUTE DATA FOR VERTICAL LEVEL

This subroutine computes level parameters at a given vertical level in the coordinate system specified by IVCORD. If VLEV is not in the data set the data will be interpolated. The output parameters must be defined by a call to PC_DFLV before this subroutine is called.

PC_CMVR (VLEV, IVCORD, DATAIN, OUTDAT, CHRDAT, IRET)

Input parameters:

VLEV	REAL	Vertical level
IVCORD	INTEGER	Vertical coordinate of VLEV 0 = NONE 1 = PRES 2 = THTA 3 = HGHT
DATAIN (NPARM,NLEV)	REAL	Station data

Output parameters:

OUTDAT (NOUTPM)	REAL	Computed real data
CHRDAT (NOUTPM)	CHAR*	Computed character data
IRET	INTEGER	Return code 0 = normal return -4 = PC_INIT must be called -6 = PC_SSTN must be called -7 = PC_DFLV must be called -10 = vert cord of dset is 0 -13 = no comp for ivcord

17.4 PC_CMVS - COMPUTE DATA FOR LEVEL NUMBER

This subroutine computes level and station data at a single vertical level specified by the level value and vertical coordinate. If the vertical level is not in the data set, the data will be interpolated. PC_DFLS should be called to define the output parameters before this subroutine is called.

PC_CMVS (VLEV, IVCORD, DATAIN, OUTDAT, CHRDAT, IRET)

Input parameters:

VLEV	REAL	Vertical level
IVCORD	INTEGER	Vertical coordinate of VLEV 0 = NONE 1 = PRES 2 = THTA 3 = HGHT
DATAIN (NPARM,NLEV)	REAL	Station data

Output parameters:

OUTDAT (NOUTPM)	REAL	Computed real data
CHRDAT (NOUTPM)	CHAR*	Computed character data
IRET	INTEGER	Return code 0 = normal return -4 = PC_INIT must be called -6 = PC_SSTN must be called -7 = output must be defined -10 = ivert = 0; vlev <> 0 -13 = no comp for ivcord

17.5 PC_DFLS - DEFINE STATION AND LEVEL PARMS

This subroutine is used to define the level and station output parameters which will be returned when the subroutine PC_CMVS is called. PC_INIT must be called before this subroutine is called. CMPFLG will be set if a parameter is computable. CHRFLG and LEVFLG will be set for character data type and station type parameters, respectively.

```
PC_DFLS ( NOUTPM, OUTPRM, CHRFLG, CMPFLG, LEVFLG, NCOMP, IRET )

Input parameters:
   NOUTPM              INTEGER         Number of output parameters
   OUTPRM (NOUTPM)     CHAR*4          Output parameters

Input and output parameters:
   CHRFLG (NOUTPM)     LOGICAL         Character data flag
   CMPFLG (NOUTPM)     LOGICAL         Computable data flag
   LEVFLG (NOUTPM)     LOGICAL         Level parameter flag
   NCOMP               INTEGER         Number of computable parms
   IRET                INTEGER         Return code
                                          0 = normal completion
                                         -4 = PC_INIT not called
                                         -5 = invalid # of parameters
```

17.6 PC_DFLV - DEFINE COMPUTED LEVEL PARAMETERS

This subroutine defines the output level parameters which will be returned when either PC_CMLV or PC_CMVR is called. The output values will be computed from the parameters in the data set. PC_INIT must be called before PC_DFLV. The returned values of CMPFLG indicate whether the parameters are computable. NCOMP is the number of computable parameters found by this subroutine.

```
PC_DFLV  ( NOUTPM, OUTPRM, CHRFLG, CMPFLG, NCOMP, IRET )

Input parameters:
   NOUTPM             INTEGER          Number of output parameters
   OUTPRM (NOUTPM)    CHAR*4           Output parameters

Output parameters:
   CHRFLG (NOUTPM)    LOGICAL          Character data flag
   CMPFLG (NOUTPM)    LOGICAL          Computable data flag
   NCOMP              INTEGER          Number of computable parms
   IRET               INTEGER          Return code
                                         0 = normal return
                                        -4 = PC_INIT not called
                                        -5 = invalid # of parms
```

17.7 PC_DFST - DEFINE COMPUTED STATION PARAMETERS

This subroutine defines the station parameters to be returned when PC_CMST is called. PC_INIT must be called to define the dataset parameters before this subroutine is called. This subroutine should only be used in programs where the station parameters will be accessed separately from the level parameters. CMPFLG indicates whether the parameters are computable.

The current station parameters are: STID, STNM, SELV, SLAT, SLON, STIM and various stability indices.

```
PC_DFST  ( NOUTPM, OUTPRM, CHRFLG, CMPFLG, NCOMP, IRET )

Input parameters:
   NOUTPM            INTEGER        Number of output parameters
   OUTPRM (NOUTPM)   CHAR*4         Output parameter names

Output parameters:
   CHRFLG (NOUTPM)   LOGICAL        Character type flag
   CMPFLG (NOUTPM)   LOGICAL        Computable flag
   NCOMP             INTEGER        Number of computable parms
   IRET              INTEGER        Return code
                                       0 = normal completion
                                      -4 = PC_INIT not called
                                      -5 = invalid NOUTPM
```

17.8 PC_FLVL - FIND LEVELS IN A COORDINATE SYSTEM

This subroutine finds the level number for a vertical level in any coordinate system. RLEV returns the actual vertical level. RLEV will equal VLEV unless VLEV is 0 or -1 for surface or top level, respectively.

```
PC_FLVL  ( VLEV, IVCORD, DATAIN, RLEV, LEVEL1, LEVEL2, LEVTYP,
           IRET )
```

Input parameters:

VLEV	REAL	Vertical level
IVCORD	INTEGER	Vertical coordinate
DATAIN (NPARM,NLEV)	REAL	Station data

Output parameters:

RLEV	REAL	Vertical level
LEVEL1	INTEGER	Level at or below VLEV
LEVEL2	INTEGER	Upper level number
LEVTYP	INTEGER	Level type 1 = data at level1 2 = data between levels 1,2 3 = data below lowest level 4 = data above top level
IRET	INTEGER	Return code 0 = normal return -4 = PC_INIT not called -6 = PC_SSTN not called -9 = no surface data -10 = IVERT=0, levl < > 0 -11 = no valid data

17.9 PC_INIT - INITIALIZE PC SUBROUTINES

This subroutine initializes the parameter conversion software. Information about the current data set is saved. It must be the first PC subroutine called.

PC_INIT (IVERT, NPARM, PARMS, IRET)

Input parameters:

IVERT	INTEGER	Vertical coordinate type 0 = NONE 1 = PRES 2 = THTA 3 = HGHT
NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*4	Parameter names

Output parameters:

IRET	INTEGER	Return code 0 = normal return -1 = invalid NPARM -2 = invalid IVERT

17.10 PC_INTP - INTERPOLATE BETWEEN TWO LEVELS

This subroutine interpolates between two levels of data. The data are interpolated with respect to the log of the pressure. Pressure MUST be the first variable in the input data arrays. If errors are encountered no output data are changed. Therefore, data should be set to RMISSD before calling this subroutine.

```
PC_INTP  ( VLEV, ADATA, BDATA, NPARMS, INTFLG, ANGFLG,
           OUTDAT, IRET )
```

Input parameters:

VLEV	REAL	Pressure to be interpolated
ADATA (NPARMS)	REAL	Data at first level
BDATA (NPARMS)	REAL	Data at second level
NPARMS	INTEGER	Size of data arrays
INTFLG (NPARMS)	LOGICAL	Interpolation flags
ANGFLG (NPARMS)	LOGICAL	Angle interpolation flags

Output parameters:

OUTDAT (NPARMS)	REAL	Data interpolated to VLEV
IRET	INTEGER	Return code 0 = normal return -24 = vlev not between levels -28 = invalid pressure

17.11 PC_SLCD - DEFINE LEVEL CONDITIONS

This subroutine sets conditions for level parameters. It must be called after the level parameters have been defined using PC_DFLV. No checks are made to verify that the conditions are valid. This subroutine may also be used to set both level and station conditions for parameters defined by calling PC_DFLS.

PC_SLCD (NOUTPM, CONDTN, IRET)

Input parameters:

NOUTPM	INTEGER	Number of output parameters
CONDTN (NOUTPM)	CHAR*	Conditions

Output parameters:

IRET	INTEGER	Return code 0 = normal return -15 = NOUTPM incorrect

17.12 PC_SSCD - DEFINE STATION CONDITIONS

This subroutine sets conditions for station parameters. It must be called after the station parameters have been defined using PC_DFST. No checks are made to verify that the conditions are valid.

PC_SSCD (NOUTPM, CONDTN, IRET)

Input parameters:

NOUTPM	INTEGER	Number of output parameters
CONDTN (NOUTPM)	CHAR*	Conditions

Output parameters:

IRET	INTEGER	Return code 0 = normal return -15 = NOUTPM incorrect

17.13 PC_SSTN - DEFINE STATION INFORMATION

This subroutine saves the station information required for the PC package.

PC_SSTN (STID, ISNUM, SLAT, SLON, SELV, IHHMM, NLEV, IRET)

Input parameters:

STID	CHAR*4	Station identifier
ISNUM	INTEGER	Station number
SLAT	REAL	Station latitude
SLON	REAL	Station longitude
SELV	REAL	Station elevation
IHHMM	INTEGER	Station hour and minute
NLEV	INTEGER	Number of vertical levels

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = invalid NLEV

17.14 PC_STIM - SET DATE/TIME

This subroutine saves the nominal time for the station report.

PC_STIM (IHHMM, IRET)

Input parameters:

IHHMM	INTEGER	Report hour and minute

Output parameters:

IRET	INTEGER	Return code 0 = normal return

CHAPTER 18

PARAMETER ARRAY (PD) LIBRARY

PD_DDEN	DDEN from PRES, TMPC
PD_DRCT	DRCT from UX, VX
PD_DWPT	DWPT from MIXR, PRES
PD_KNMS	SPED from SKNT
PD_MIXR	MIXR from DWPC, PRES
PD_MSKN	SKNT from SPED
PD_RELH	RELH from TMPC, DWPC
PD_RHDP	DWPC from TMPC, RELH
PD_SDUV	Speed/direction to u/v
PD_SPED	SPED from UWND, VWND
PD_THTA	THTA from TMPC, PRES
PD_THTE	THTE from PRES, TMPC, DWPC
PD_TMCF	TMPF from TMPC
PD_TMCK	TMPK from TMPC
PD_TMFC	TMPC from TMPF
PD_TMFK	TMPK from TMPF
PD_TMKC	TMPC from TMPK
PD_TMKF	TMPF from TMPK
PD_TMPK	TMPK from PRES, THTA
PD_TVRK	TVRK from TMPC, DWPC, PRES
PD_UVSD	U/V to speed/direction

Parameter Array (PD) Library Summary

This parameter library contains subroutines to compute meteorological parameters for arrays of data. These subroutines are used by the grid diagnostics package to calculate parameters. They may also be called directly by application programs, provided that the programmer is careful to use the subsetting capabilities properly. In each subroutine, SUBFLG is a logical array. At each point, if SUBFLG is true, the calculation will be made; if it is false, no calculation will be done.

Note that this library is derived from the PR library. Therefore, changes made in either library should also be made in the other.

The following constants are used in the computations:

RKAPPA =	Poisson's constant	= 2 / 7
G =	Gravitational constant	= 9.80616 m/sec/sec
GAMMA =	Standard atmospheric lapse rate	= 6.5 K/km
RDGAS =	Gas constant for dry air	= 287.04 J/K/kg

Many of the conversion algorithms are taken from:

Bolton, D., 1980: The computation of equivalent potential temperature. MONTHLY WEATHER REVIEW, 108, pp. 1046-1053.

University of Wisconsin: Green sheet.

Wallace, J.M., P.V. Hobbs, 1977: ATMOSPHERIC SCIENCE, Academic Press, 467 pp.

PARAMETER ARRAY (PD) LIBRARY

PD Library Calls

PD_DDEN	(pres, tmpc, npt, subflg, / dden, iret)
PD_DRCT	(uwnd, vwnd, npt, subflg, / drct, iret)
PD_DWPT	(rmix, pres, npt, subflg, / dwpc, iret)
PD_KNMS	(sknt, npt, subflg, / sped, iret)
PD_MIXR	(dwpc, pres, npt, subflg, / rmix, iret)
PD_MSKN	(sped, npt, subflg, / sknt, iret)
PD_RELH	(tmpc, dwpc, npt, subflg, / relh, iret)
PD_RHDP	(tmpc, relh, npt, subflg, / dwpc, iret)
PD_SDUV	(sped, drct, npt, subflg, uwnd, vwnd, iret)
PD_SPED	(uwnd, vwnd, npt, subflg, / sped, iret)
PD_THTA	(tmpc, pres, npt, subflg, / thta, iret)
PD_THTE	(pres, tmpc, dwpc, npt, subflg, / thte, iret)
PD_TMCF	(tmpc, npt, subflg, / tmpf, iret)
PD_TMCK	(tmpc, npt, subflg, / tmpk, iret)
PD_TMFC	(tmpf, npt, subflg, / tmpc, iret)
PD_TMFK	(tmpf, npt, subflg, / tmpk, iret)
PD_TMKC	(tmpk, npt, subflg, / tmpc, iret)
PD_TMKF	(tmpk, npt, subflg, / tmpf, iret)
PD_TMPK	(prgrd, thgrd, npt, subflg, / tmpk, iret)
PD_TVRK	(tmpk, rmix, npt, subflg, / tvrk, iret)
PD_UVSD	(uwnd, vwnd, npt, subflg, / sped, drct, iret)

18.1 PD_DDEN - DDEN FROM PRES, TMPC

This subroutine computes DDEN from PRES, TMPC. The following equation is used:

DDEN = 100 * PRES / (RDGAS * TMPK)

100 = conversion from millibars to pascals

PD_DDEN (PRES, TMPC, NPT, SUBFLG, DDEN, IRET)

Input parameters:

PRES (NPT)	REAL	Pressure in millibars
TMPC (NPT)	REAL	Temperature in Celsius
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

DDEN (NPT)	REAL	Dry air density in kg/(m**3)
IRET	INTEGER	Return code 0 = normal return

18.2 PD_DRCT - DRCT FROM UX, VX

This subroutine computes DRCT from UWND and VWND. The following equation is used:

DRCT = ATAN2 (-UWND, -VWND) * RTD

PD_DRCT (UWND, VWND, NPT, SUBFLG, DRCT, IRET)

Input parameters:

UWND (NPT)	REAL	U component
VWND (NPT)	REAL	V component
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

DRCT (NPT)	REAL	Wind direction
IRET	INTEGER	Return code 0 = normal return

18.3 PD_DWPT - DWPT FROM MIXR, PRES

This subroutine computes DWPC from MIXR and PRES. The following equation is used:

```
  DWPC = ALOG (E / 6.112) * 243.5 / ( 17.67 - ALOG (E / 6.112) )

         E = vapor pressure
           = e / ( 1.001 + ( (PRES - 100.) / 900. ) * .0034 )
         e = ( PRES * MIXR ) / ( .62197 + MIXR )
```

Bolton.

This subroutine also computes TMPC from MIXS and PRES.

PD_MIXR (RMIX, PRES, NPT, SUBFLG, DWPC, IRET)

Input parameters:

RMIX (NPT)	REAL	Mixing ratio in g/kg
PRES (NPT)	REAL	Pressure in millibars
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

DWPC (NPT)	REAL	Dewpoint in Celsius
IRET	INTEGER	Return code 0 = normal return

18.4 PD_KNMS - SPED FROM SKNT

This subroutine computes SPED from SKNT. The following equation is used:

$$SPED = SKNT / 1.9438$$

PD_KNMS (SKNT, NPT, SUBFLG, SPED, IRET)

Input parameters:

SKNT (NPT)	REAL	Speed in knots
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

SPED (NPT)	REAL	Speed in meters/second
IRET	INTEGER	Return code 0 = normal return

18.5 PD_MIXR - MIXR FROM DWPC, PRES

This subroutine computes MIXR from DWPC and PRES. The following equation is used:

```
MIXR = .62197 * ( e / ( PRES - e ) ) * 1000.

     e    =  VAPR * corr
     corr =  (1.001 + ( ( PRES - 100. ) / 900. ) * .0034)
```

This function can also be used for the following computations:

```
MIXS from TMPC and PRES
SMXR from DWPC and PALT
SMXS from TMPC and PALT
```

PD_MIXR (DWPC, PRES, NPT, SUBFLG, RMIX, IRET)

Input parameters:

DWPC (NPT)	REAL	Dewpoint in Celsius
PRES (NPT)	REAL	Pressure in millibars
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

RMIX (NPT)	REAL	Mixing ratio in g/kg
IRET	INTEGER	Return code 0 = normal return

18.6 PD_MSKN - SKNT FROM SPED

This subroutine computes SKNT from SPED. The following equation is used:

SKNT = SPED * 1.9438

PD_MSKN (SPED, NPT, SUBFLG, SKNT, IRET)

Input parameters:

SPED (NPT)	REAL	Speed in meters/second
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

SKNT (NPT)	REAL	Speed in knots
IRET	INTEGER	Return code 0 = normal return

18.7 PD_RELH - RELH FROM TMPC, DWPC

This subroutine computes RELH from TMPC and DWPC. The following equation is used:

```
RELH  =  VAPR / VAPS * 100

         VAPR = vapor pressure
              = PR_VAPR ( DWPC )
         VAPS = saturation vapor pressure
              = PR_VAPR ( TMPC )
```

PD_RELH (TMPC, DWPC, NPT, SUBFLG, RELH, IRET)

Input parameters:

TMPC (NPT)	REAL	Temperature in Celsius
DWPC (NPT)	REAL	Dewpoint in Celsius
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

RELH (NPT)	REAL	Relative humidity in percent
IRET	INTEGER	Return code 0 = normal return

18.8 PD_RHDP - DWPC FROM TMPC, RELH

This subroutine computes DWPC from TMPC and RELH. The following equation is used:

```
DWPC = 243.5 * LN (6.112) - 243.5 * LN (VAPR) /
         ( LN (VAPR) - LN (6.112) - 17.67 )

     VAPR = VAPS * RELH
     VAPS = saturation vapor pressure
          = PR_VAPR ( TMPC )
```

Note: If DWPC is less than -190 degrees C, it is treated as missing data.

PD_RHDP (TMPC, RELH, NPT, SUBFLG, DWPC, IRET)

Input parameters:

TMPC (NPT)	REAL	Temperature in Celsius
RELH (NPT)	REAL	Relative humidity in percent
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

DWPC (NPT)	REAL	Dewpoint in Celsius
IRET	INTEGER	Return code 0 = normal return

18.9 PD_SDUV - SPEED/DIRECTION TO U/V

This subroutine computes UWND and VWND from SPED and DRCT for an array. The following equations are used:

UWND = -SIN (DRCT) * SPED
VWND = -COS (DRCT) * SPED

PD_SDUV (SPED, DRCT, NPT, SUBFLG, UWND, VWND, IRET)

Input parameters:

SPED (NPT)	REAL	Wind speed
DRCT (NPT)	REAL	Wind direction in degrees
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

UVND (NPT)	REAL	U component
VWND (NPT)	REAL	V component
IRET	INTEGER	Return code 0 = normal return

18.10 PD_SPED - SPED FROM UWND, VWND

This subroutine computes SPED from UWND and VWND. The following equation is used:

```
SPED = SQRT ( (UWND**2) + (VWND**2) )
```

```
PD_SPED ( UWND, VWND, NPT, SUBFLG, SPED, IRET )
```

Input parameters:

UWND (NPT)	REAL	U component
VWND (NPT)	REAL	V component
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

SPED (NPT)	REAL	Wind speed
IRET	INTEGER	Return code 0 = normal return

18.11 PD_THTA - THTA FROM TMPC, PRES

This subroutine computes THTA from TMPC and PRES using Poisson's equation:

THTA = TMPK * (1000 / PRES) ** RKAPPA

It can also be used to compute STHA from TMPC and PALT, THTV from TVRC and PRES, and THTV from TVRC and PALT.

PD_THTA (TMPC, PRES, NPT, SUBFLG, THTA, IRET)

Input parameters:

TMPC (NPT)	REAL	Temperature in Celsius
PRES (NPT)	REAL	Pressure in millibars
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

THTA (NPT)	REAL	Potential temperature in K
IRET	INTEGER	Return code 0 = normal return

18.12 PD_THTE - THTE FROM PRES, TMPC, DWPC

This subroutine computes THTE from PRES, TMPC, DWPC. In the calculation, MIXR depends on PRES and DWPC; TLCL depends on TMPC and DWPC. The following equation is used:

```
THTE = THTAM * EXP [ ( 3.376/TLCL - .00254 ) *
                     ( MIXR * ( 1 + .81*.001*MIXR ) ) ]

        THTAM = potential temperature of moist air
              = TMPK * (1000 / PRES) ** E
            E = RKAPPA * ( 1 - ( .28 * .001 * MIXR ) )
```

PD_THTE (PRES, TMPC, DWPC, NPT, SUBFLG, THTE, IRET)

Input parameters:

PRES (NPT)	REAL	Pressure in millibars
TMPC (NPT)	REAL	Temperature in Celsius
DWPC (NPT)	REAL	Dewpoint in Celsius
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

THTE (NPT)	REAL	Equivalent potential temp in K
IRET	INTEGER	Return code 0 = normal return

18.13 PD_TMCF - TMPF FROM TMPC

This subroutine computes TMPF from TMPC. The following equation is used:

```
TMPF = ( TMPC * 9 / 5 ) + 32
```

```
PD_TMCF ( TMPC, NPT, SUBFLG, TMPF, IRET )

Input parameters:
   TMPC (NPT)        REAL           Temperature in Celsius
   NPT               INTEGER        Number of points
   SUBFLG (NPT)      LOGICAL        Subset flag

Output parameters:
   TMPF (NPT)        REAL           Temperature in Fahrenheit
   IRET              INTEGER        Return code
                                      0 = normal return
```

18.14 PD_TMCK - TMPK FROM TMPC

This subroutine computes TMPK from TMPC. The following equation is used:

TMPK = TMPC + 273.16

PD_TMCK (TMPC, NPT, SUBFLG, TMPK, IRET)

Input parameters:

TMPC (NPT)	REAL	Temperature in Celsius
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

TMPK (NPT)	REAL	Temperature in Kelvin
IRET	INTEGER	Return code 0 = normal return

18.15 PD_TMFC - TMPC FROM TMPF

This subroutine computes TMPC from TMPF. The following equation is used:

TMPC = (TMPF - 32) * 5 / 9

PD_TMFC (TMPF, NPT, SUBFLG, TMPC, IRET)

Input parameters:

TMPF (NPT)	REAL	Temperature in Fahrenheit
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

TMPC (NPT)	REAL	Temperature in Celsius
IRET	INTEGER	Return code 0 = normal return

18.16 PD_TMFK - TMPK FROM TMPF

This subroutine computes TMPK from TMPF. The following equation is used:

TMPK = (TMPF - 32) * 5 / 9 + 273.16

PD_TMFK (TMPF, NPT, SUBFLG, TMPK, IRET)

Input parameters:

TMPF (NPT)	REAL	Temperature in Fahrenheit
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

TMPK (NPT)	REAL	Temperature in Kelvin
IRET	INTEGER	Return code 0 = normal return

18.17 PD_TMKC - TMPC FROM TMPK

This subroutine computes TMPC from TMPK. The following equation is used:

$$TMPC = TMPK - 273.16$$

PD_TMKC (TMPK, NPT, SUBFLG, TMPC, IRET)

Input parameters:

TMPK (NPT)	REAL	Temperature in Kelvin
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

TMPC (NPT)	REAL	Temperature in Celsius
IRET	INTEGER	Return code 0 = normal return

18.18 PD_TMKF - TMPF FROM TMPK

This subroutine computes TMPF from TMPK. The following equation is used:

TMPF = ((TMPK - 273.16) * 9 / 5) + 32

PD_TMKF (TMPK, NPT, SUBFLG, TMPF, IRET)

Input parameters:

TMPK (NPT)	REAL	Temperature in Kelvin
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

TMPF (NPT)	REAL	Temperature in Fahrenheit
IRET	INTEGER	Return code 0 = normal return

18.19 PD_TMPK - TMPK FROM PRES, THTA

This subroutine computes TMPK from pressure and potential temperature. The following equation is used:

```
TMPK = THTA * ( P / 1000 ) ** RKAPPA
```

```
PD_TMPK  ( PRGRD, THGRD, NPT, SUBFLG, TMPK, IRET )

Input parameters:
   PRGRD (NPT)       REAL              Pressure in mb
   THGRD (NPT)       REAL              THETA in Kelvin
   NPT               INTEGER           Number of points
   SUBFLG (NPT)      LOGICAL           Subset flag

Output parameters:
   TMPK (NPT)        REAL              Temperature in Kelvin
   IRET              INTEGER           Return code
                                         0 = normal return
```

18.20 PD_TVRK - TVRK FROM TMPC, DWPC, PRES

This subroutine computes TVRK from TMPK, RMIX. The following equation is used:

TVRK = TMPK * (1 + .001 * MIXR / .62197) / (1 + .001 * MIXR)

Note that this subroutine requires different inputs than the function PR_TVRK.

PD_TVRK (TMPK, RMIX, NPT, SUBFLG, TVRK, IRET)

Input parameters:

TMPK (NPT)	REAL	Temperature in Kelvin
RMIX (NPT)	REAL	Mixing ratio in g/g
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

TVRK (NPT)	REAL	Virtual temp in Kelvin
IRET	INTEGER	Return code 0 = normal return

18.21 PD_UVSD - U/V TO SPEED/DIRECTION

This subroutine computes SPED and DRCT from UWND and VWND. The following equations are used:

```
SPED = SQRT ( (UWND**2) + (VWND**2) )
DRCT = ATAN2 ( -UWND, -VWND ) * RTD
```

The input and output arrays may be the same.

PD_UVSD (UWND, VWND, NPT, SUBFLG, SPED, DRCT, IRET)

Input parameters:

UWND (NPT)	REAL	U component
VWND (NPT)	REAL	V component
NPT	INTEGER	Number of points
SUBFLG (NPT)	LOGICAL	Subset flag

Output parameters:

SPED (NPT)	REAL	Wind speed
DRCT (NPT)	REAL	Wind direction in degrees
IRET	INTEGER	Return code 0 = normal return

CHAPTER 19

PARAMETER (PR) LIBRARY

PR_6SYM	WWMO from W604
PR_ALTI	ALTI from ALTM
PR_ALTM	ALTM from ALTI
PR_ALTP	ALTI from PRES, SELV
PR_AMSL	SALT from ALTM
PR_CLCT	CLCT from CLCL, CLCM, CLCH
PR_CLCX	CLCX from COMX
PR_CLHX	CLHX from COMX
PR_CLOA	XCLO from CLCX
PR_CMBC	CMBC from CLCL, CLCM, CLCH
PR_COMH	COMH from CHC1, CHC2, CHC3
PR_COML	COML from CHC1, CHC2, CHC3
PR_COMM	COMM from CHC1, CHC2, CHC3
PR_COMT	COMT from COML, COMM, COMH
PR_COMX	COMX from CLHX, CLCX
PR_D100	Divide by 100
PR_DDEN	DDEN from PRES, TMPC
PR_DDEP	DPDX from TMPX, DWPX
PR_DRCT	DRCT from UX, VX
PR_DWDP	DWPX from TMPX, DPDX
PR_DWPT	DWPT from MIXR, PRES
PR_HGFM	HGHT from HGFT
PR_HGFS	HGML from HGFT
PR_HGKM	HGHT from HGTK
PR_HGMD	HGTD from HGHT
PR_HGMF	HGFT from HGHT
PR_HGMK	HGTK from HGHT
PR_HGSF	HGFT from HGML
PR_INMM	MM from INCHES/100
PR_KNMS	SPED from SKNT
PR_LATI	LATI from RANG, AZIM
PR_LHVP	LHVP from TMPC
PR_LONI	LONI from RANG, AZIM
PR_LTMP	Lifted parcel
PR_M100	Multiply by 100

PARAMETER (PR) LIBRARY

PR_MHGT	Compute MHGT
PR_MIXR	MIXR from DWPC, PRES
PR_MMIN	INCHES/100 from MM
PR_MSKN	SKNT from SPED
PR_NSYM	WWMO from WNUM
PR_PALT	PALT from ALTM, SELV
PR_PKDD	DRCT from PSPD
PR_PKSS	SPED from PSPD
PR_PLCL	PLCL from TMPC, PRES, TLCL
PR_PMSL	PMSL from PRES, TMPC, DWPC, SELV
PR_PRES	PRES from TMPC, THTA
PR_PSPD	PSPD from DRCT, SPED
PR_RELH	RELH from TMPC, DWPC
PR_RHDP	DWPC from TMPC, RELH
PR_RWSH	RWSH from Asheville code
PR_SALI	SALI from ALTI
PR_SCLH	Compute scale height
PR_SPED	SPED from UWND, VWND
PR_STDZ	STDZ from PRES, HGHT
PR_THTA	THTA from TMPC, PRES
PR_THTE	THTE from PRES, TMPC, DWPC
PR_TLCL	TLCL from TMPC, DWPC
PR_TMCF	TMPF from TMPC
PR_TMCK	TMPK from TMPC
PR_TMFC	TMPC from TMPF
PR_TMFK	TMPK from TMPF
PR_TMKC	TMPC from TMPK
PR_TMKF	TMPF from TMPK
PR_TMPK	TMPK from PRES, THTA
PR_TMST	Parcel temperature
PR_TVRK	TVRK from TMPC, DWPC, PRES
PR_UWND	UWND from SPED, DRCT
PR_VAPR	VAPR from DWPC
PR_VWND	VWND from SPED, DRCT
PR_WCMP	Wind component
PR_WNML	Normal wind component
PR_ZALT	ZALT from ALTM, PRES

PARAMETER (PR) LIBRARY

Parameter (PR) Library Summary

The parameter library contains functions to compute meteorological parameters. These functions are used by the PC subroutine package to convert parameters and may also be called directly by application programs.

The library was designed to provide easy access to standard meteorological conversions. Any desired change to a parameter conversion function, either to add precision or make a correction, which is made to a PR function will then be effective throughout GEMPAK when the executable code is relinked.

The following constants are used in the computations:

RKAPPA - Poisson's constant		= 2 / 7
G - Gravitational constant		= 9.80616 m/sec/sec
GAMMA - Standard atmospheric lapse rate		= 6.5 K/km
RDGAS - Gas constant for dry air		= 287.04 J/K/kg

Many of the conversion algorithms are taken from:

Bolton, D., 1980: The computation of equivalent potential temperature. MONTHLY WEATHER REVIEW, 108, pp. 1046-1053.

University of Wisconsin: Green sheet.

Wallace, J.M., P.V. Hobbs, 1977: ATMOSPHERIC SCIENCE, Academic Press, 467 pp.

PR Library Calls

PR_6SYM (w604)

PR_ALTI (altm)

PR_ALTM (alti)

PR_ALTP (pres, selv)

PR_AMSL (pmsl)

PR_CLCT (clcl, clcm, clch)

PR_CLCX (comx)

PR_CLHX (comx)

PR_CLOA (clcx)

PR_CMBC (clcl, clcm, clch)

PR_COMH (chc1, chc2, chc3)

PR_COML (chc1, chc2, chc3)

PR_COMM (chc1, chc2, chc3)

PR_COMT (coml, comm, comh)

PR_COMX (clhx, clcx)

PR_D100 (hvalue)

PR_DDEN (pres, tmpc)

PR_DDEP (tmpx, dwpx)

PR_DRCT (ux, vx)

PR_DWDP (tmpx, dpdx)

PR_DWPT (rmix, pres)

PR_HGFM (hgft)

PR_HGFS (hgft)

PR_HGKM (value)

PR_HGMD (hght)

PR_HGMF	(hght)
PR_HGMK	(value)
PR_HGSF	(hgml)
PR_INMM	(xinch)
PR_KNMS	(sknt)
PR_LATI	(slat, slon, range, azim, selv)
PR_LHVP	(tmpc)
PR_LONI	(slat, slon, range, azim, selv)
PR_LTMP	(thta, thte, pres)
PR_M100	(value)
PR_MHGT	(hb, pb, pt, scale)
PR_MIXR	(dwpc, pres)
PR_MMIN	(xmilm)
PR_MSKN	(sped)
PR_NSYM	(wnum)
PR_PALT	(altm, selv)
PR_PKDD	(pspd)
PR_PKSS	(pspd)
PR_PLCL	(tmpc, pres, tlcl)
PR_PMSL	(pres, tmpc, dwpc, selv)
PR_PRES	(tmpc, thta)
PR_PSPD	(drct, sped)
PR_RELH	(tmpc, dwpc)
PR_RHDP	(tmpc, relh)
PR_RWSH	(inum)

```
PR_SALI    ( alti )
PR_SCLH    ( tb, tt, tdb, tdt, pb, pt )
PR_SPED    ( uwnd, vwnd )
PR_STDZ    ( pres, hght )
PR_THTA    ( tmpc, pres )
PR_THTE    ( pres, tmpc, dwpc )
PR_TLCL    ( tmpc, dwpc )
PR_TMCF    ( tmpc )
PR_TMCK    ( tmpc )
PR_TMFC    ( tmpf )
PR_TMFK    ( tmpf )
PR_TMKC    ( tmpk )
PR_TMKF    ( tmpk )
PR_TMPK    ( pres, thta )
PR_TMST    ( thte, pres, tguess )
PR_TVRK    ( tmpc, dwpc, pres )
PR_UWND    ( sped, drct )
PR_VAPR    ( dwpc )
PR_VWND    ( sped, drct )
PR_ZALT    ( altm, pres )
```

19.1 PR_6SYM - WWMO FROM W604

This function converts the airways code W604 to the WMO weather code, WWMO, which is used to plot weather symbols.

PR_6SYM (W604)

Input parameters:

W604	REAL	Airways 604 numeric code

Output parameters:

PR_6SYM	REAL	Weather symbol number

19.2 PR_ALTI - ALTI FROM ALTM

This function computes ALTI from ALTM. The following equation is used:

ALTI = ALTM * 29.921 / 1013.25

PR_ALTI (ALTM)

Input parameters:

ALTM	REAL	Altimeter in millibars

Output parameters:

PR_ALTI	REAL	Altimeter in inches

19.3 PR_ALTM - ALTM FROM ALTI

This function computes ALTM from ALTI. The following equation is used:

$$ALTM = ALTI * 1013.25 / 29.921$$

PR_ALTM (ALTI)

Input parameters:

ALTI	REAL	Altimeter in inches

Output parameters:

PR_ALTM	REAL	Altimeter in millibars

19.4 PR_ALTP - ALTI FROM PRES, SELV

This function computes ALTI from PRES and SELV. The following equation is used:

```
ALTI = PR_ALTI ( PRES * ( 1 - ( SELK * GAMUSD / To ) ) ** expo )

        SELK  =  SELV / 1000
          To  =  US Std. Atmos. sea level temp in Kelvin
              =  288.
        expo  =  - GRAVTY / ( GAMUSD * RDGAS ) * 1000
```

Wallace and Hobbs

PR_ALTP (PRES, SELV)

Input parameters:

PRES	REAL	Station pressure in millibars
SELV	REAL	Station elevation in meters

Output parameters:

PR_ALTP	REAL	Altimeter in inches

19.5 PR_AMSL - SALT FROM ALTM

This function computes a standard abbreviated 3-digit display of pressure containing the tens and units digits and the first digit after the decimal point. The input is multiplied by 10, truncated, and the original thousand and hundred digits are dropped. The following equation is used:

```
AMSL = NINT ( AMOD ( PMSL, 100 ) * 10 )
```

This function can be used to compute SALT from ALTM, and SMSL from PMSL.

```
PR_AMSL ( PMSL )
```

Input parameters:

PMSL	REAL	Pressure in mb

Output parameters:

PR_AMSL	REAL	Standard abbreviated pressure

19.6 PR_CLCT - CLCT FROM CLCL, CLCM, CLCH

This function computes CLCT from CLCL, CLCM, and CLCH. CLCT is the maximum cloud coverage of CLCL, CLCM, and CLCH, with coverage values ordered as follows:

0	no clouds
1	clear
6	thin scattered
2	scattered
7	thin broken
3	broken
8	thin overcast
4	overcast
9	partially obscured
5	obscured

PR_CLCT (CLCL, CLCM, CLCH)

Input parameters:

CLCL	REAL	Low cloud cover
CLCM	REAL	Medium cloud cover
CLCH	REAL	High cloud cover

Output parameters:

PR_CLCT	REAL	Maximum cloud cover

19.7 PR_CLCX - CLCX FROM COMX

This function gets CLCx from COMx, where x represents the L (Low), M (Mid), or H (High) cloud level. COMX is the cloud height (in hundreds of feet) * 10 + the numeric cloud coverage code.

Input parameters:

COMX	REAL	Combined height and coverage

Output parameters:

PR_CLCX	REAL	Numeric cloud coverage code

19.8 PR_CLHX - CLHX FROM COMX

This function gets CLHx from COMx, where x represents the L (Low), M (Mid), or H (High) cloud level. COMx is the cloud height (in hundreds of feet) * 10 + the numeric cloud coverage code.

Input parameters:

COMX	REAL	Combined height and coverage

Output parameters:

PR_CLHX	REAL	Cloud height in feet * 100

19.9 PR_CLOA - XCLO FROM CLCX

This function computes XCLO from CLCx. The output is the fractional cloud coverage; x may be L, M, H or T. The cloud coverage values and the corresponding fractional equivalents are:

CLCx	CLOUD TYPE	XCLO
0	no clouds	0.000
1	clear	0.000
2	scattered	0.333
3	broken	0.667
4	overcast	1.000
5	obscured	1.000
6	thin scatterd	0.167
7	thin broken	0.500
8	thin overcast	0.833
9	partially obscured	1.000

PR_CLOA (CLCX)

Input parameters:

CLCX	REAL	Numeric cloud coverage

Output parameters:

PR_CLOA	REAL	Fractional cloud coverage

19.10 PR_CMBC - CMBC FROM CLCL, CLCM, CLCH

This function computes CMBC, the combined low, mid and high cloud coverages, from CLCL, CLCM, and CLCH. The following equation is used:

CMBC = (CLCL * 100) + (CLCM * 10) + CLCH

PR_CMBC (CLCL, CLCM, CLCH)

Input parameters:

CLCL	REAL	Low cloud coverage
CLCM	REAL	Medium cloud coverage
CLCH	REAL	High cloud coverage

Output parameters:

PR_CMBC	REAL	Combined cloud coverage

19.11 PR_COMH - COMH FROM CHC1, CHC2, CHC3

This function gets COMH from CHC1, CHC2, and CHC3. COMH is the combined height and numeric sky coverage for high clouds which are those whose height is greater than 179,000 feet.

PR_COMH (CHC1, CHC2, CHC3)

Input parameters:

CHC1	REAL	Cloud height & coverage 1
CHC2	REAL	Cloud height & coverage 2
CHC3	REAL	Cloud height & coverage 3

Output parameters:

PR_COMH	REAL	Hi combined height & coverage

19.12 PR_COML - COML FROM CHC1, CHC2, CHC3

This function gets COML from CHC1, CHC2, and CHC3. COML is the combined height and numeric sky coverage for low clouds which are those whose height is less than 63,000 feet.

PR_COML (CHC1, CHC2, CHC3)

Input parameters:

CHC1	REAL	Cloud height & coverage 1
CHC2	REAL	Cloud height & coverage 2
CHC3	REAL	Cloud height & coverage 3

Output parameters:

PR_COML	REAL	Low combined height & coverage

19.13 PR_COMM - COMM FROM CHC1, CHC2, CHC3

This function gets COMM from CHC1, CHC2, and CHC3. COMM is the combined height and numeric sky coverage for medium clouds which are those whose height is greater than 63,000 feet and less than 179,000 feet.

PR_COMM (CHC1, CHC2, CHC3)

Input parameters:

CHC1	REAL	Cloud height & coverage 1
CHC2	REAL	Cloud height & coverage 2
CHC3	REAL	Cloud height & coverage 3

Output parameters:

PR_COMM	REAL	Med combined height & coverage

19.14 PR_COMT - COMT FROM COML, COMM, COMH

This function computes COMT from COML,COMM and COMH.

PR_COMT (COML, COMM, COMH)

Input parameters:

COML	REAL	Low report height & coverage
COMM	REAL	Mid report height & coverage
COMH	REAL	High report height & coverage

Output parameters:

PR_COMT	REAL	Highest combined height & coverage

19.15 PR_COMX - COMX FROM CLHX, CLCX

This function computes COMX, the combined cloud height and coverage, from CLHX and CLCX. The following equation is used:

COMX = (CLHX * 10) + CLCX

PR_COMX (CLHX, CLCX)

Input parameters:

CLHX	REAL	Cloud height
CLCX	REAL	Cloud coverage

Output parameters:

PR_COMX	REAL	Combined height & coverage

19.16 PR_D100 - DIVIDE BY 100

This function divides a value by 100.

PR_D100 (VALUE)

Input parameters:

VALUE	REAL	Value

Output parameters:

PR_D100	REAL	Value / 100

19.17 PR_DDEN - DDEN FROM PRES, TMPC

This function computes DDEN from PRES, TMPC. The following equation is used:

DDEN = 100 * PRES / (RDGAS * TMPK)

100: conversion from millibars to pascals

PR_DDEN (PRES, TMPC)

Input parameters:

PRES	REAL	Pressure in millibars
TMPC	REAL	Temperature in Celsius

Output parameters:

PR_DDEN	REAL	Density of dry air in kg/(m**3)

19.18 PR_DDEP - DPDX FROM TMPX, DWPX

This function computes DPDX, the dewpoint depression, from TMPX and DWPX, both of which must be in the same units (Celsius, Kelvin, or Fahrenheit). The output will be calculated in these units. The following equation is used:

DPDX = TMPX - DWPX

PR_DDEP (TMPX, DWPX)

Input parameters:

TMPX	REAL	Air temperature
DWPX	REAL	Dewpoint temperature

Output parameters:

PR_DDEP	REAL	Dewpoint depression

19.19 PR_DRCT - DRCT FROM UX, VX

This function computes DRCT from Ux and Vx, both of which must be either meters/sec or knots. The following equation is used:

DRCT = ATAN2 (-UX, -VX) * RTD

PR_DRCT (UX, VX)

Input parameters:

UX	REAL	U component of velocity
VX	REAL	V component of velocity

Output parameters:

PR_DRCT	REAL	Wind direction in degrees

19.20 PR_DWDP - DWPX FROM TMPX, DPDX

This function computes DWPX from TMPX and DPDX, both of which must be in the same units (Celsius, Kelvin, or Fahrenheit). DWPX will be calculated in these units. The following equation is used:

DWPX = TMPX - DPDX

PR_DWDP (TMPX, DPDX)

Input parameters:

TMPX	REAL	Temperature
DPDX	REAL	Dewpoint depression

Output parameters:

PR_DWDP	REAL	Dewpoint

19.21 PR_DWPT - DWPT FROM MIXR, PRES

This function computes DWPT from MIXR and PRES. The following equation is used:

```
  DWPT = ALOG (E / 6.112) * 243.5 / ( 17.67 - ALOG (E / 6.112) )

       E = vapor pressure
         = e / ( 1.001 + ( (PRES - 100.) / 900. ) * .0034 )
       e = ( PRES * MIXR ) / ( .62197 + MIXR )
```

Bolton.

This function also computes TMPC from MIXS and PRES.

PR_DWPT (MIXR, PRES)

Input parameters:

MIXR	REAL	Mixing ratio in g/kg
PRES	REAL	Pressure in millibars

Output parameters:

PR_DWPT	REAL	Dewpoint in Celsius

19.22 PR_HGFM - HGHT FROM HGFT

This function computes HGHT from HGFT. The following equation is used:

HGHT = HGFT * .3048

PR_HGFM (HGFT)

Input parameters:

HGFT	REAL	Height in feet

Output parameters:

PR_HGFM	REAL	Height in meters

19.23 PR_HGFS - HGML FROM HGFT

This function computes HGML, height in miles, from HGFT. The following equation is used:

$$HGML = HGFT / 5280.$$

PR_HGFS (HGFT)

Input parameters:

HGFT	REAL	Height in feet

Output parameters:

PR_HGFS	REAL	Height in statute miles

19.24 PR_HGKM - HGHT FROM HGTK

This function multiplies a value by a thousand. It can be used to compute meters from kilometers.

PR_HGKM (VALUE)

Input parameters:

VALUE	REAL	Value

Output parameters:

PR_HGKM	REAL	Value * 1000

19.25 PR_HGMD - HGTD FROM HGHT

This function computes HGTD from HGHT. The following equation is used:

$$HGTD = HGHT / 10.$$

PR_HGMD (HGHT)

Input parameters:

HGHT	REAL	Height in meters

Output parameters:

PR_HGMD	REAL	Height in decameters

19.26 PR_HGMF - HGFT FROM HGHT

This function computes HGFT from HGHT. The following equation is used:

HGFT = NINT (HGHT * 3.28084)

(Note: This function rounds to the nearest foot.)

PR_HGMF (HGHT)

Input parameters:

HGHT	REAL	Height in meters

Output parameters:

PR_HGMF	REAL	Height in feet

19.27 PR_HGMK - HGTK FROM HGHT

This function divides a value by 1000. It can be used to convert meters to kilometers.

PR_HGMK (VALUE)

Input parameters:

VALUE	REAL	Value

Output parameters:

PR_HGMK	REAL	Value / 1000

19.28 PR_HGSF - HGFT FROM HGML

This function computes HGFT from HGML. The following equation is used:

HGFT = HGML * 5280

PR_HGSF (HGML)

Input parameters:

HGML	REAL	Height in statute miles

Output parameters:

PR_HGSF	REAL	Height in feet

19.29 PR_INMM - MM FROM INCHES/100

This function converts hundredths of inches to millimeters. The following equation is used:

INMM = XINCH * .254

PR_MMIN (XINCH)

Input parameters:

XINCH	REAL	Hundredths of inches

Output parameters:

PR_INMM	REAL	Millimeters

19.30 PR_KNMS - SPED FROM SKNT

This function computes SPED from SKNT. The following equation is used:

$$SPED = SKNT / 1.9438$$

PR_KNMS (SKNT)

Input parameters:

SKNT	REAL	Speed in knots

Output parameters:

PR_KNMS	REAL	Speed in meters/second

19.31 PR_LATI - LATI FROM RANG, AZIM

This function computes LATI given the range, azimuth and station latitude, longitude and elevation. Equations developed for use in the AOIPS radar package are used.

PR_LATI (SLAT, SLON, RANGE, AZIM, SELV)

Input parameters:

SLAT	REAL	Station latitude
SLON	REAL	Station longitude
RANGE	REAL	Range in kilometers
AZIM	REAL	Geographic azimuth in radians
SELV	REAL	Station elevation

Output parameters:

PR_LATI	REAL	Actual latitude

19.32 PR_LHVP - LHVP FROM TMPC

This function computes LHVP from TMPC. LHVP, the latent heat of vaporization at constant pressure, is computed as follows:

LHVP = (2.501 - .00237 * TMPC) * 10E6

PR_LHVP (TMPC)

Input parameters:

TMPC	REAL	Temperature in Celsius

Output parameters:

PR_LHVP	REAL	Latent heat in J/kg

19.33 PR_LONI - LONI FROM RANG, AZIM

This function computes LONI given the range, azimuth and station latitude, longitude and elevation. Equations developed for use in the AOIPS radar package are used.

PR_LONI (SLAT, SLON, RANGE, AZIM, SELV)

Input parameters:

SLAT	REAL	Station latitude
SLON	REAL	Station longitude
RANGE	REAL	Range in kilometers
AZIM	REAL	Geographic azimuth in radians
SELV	REAL	Station elevation

Output parameters:

PR_LONI	REAL	Actual longitude

19.34 PR_LTMP - LIFTED PARCEL

This function computes the temperature of a parcel lifted (or sunk) adiabatically to a given pressure.

PR_LTMP (THTA, THTE, PRES)

Input parameters:

THTA	REAL	Potential temperature in Kelvin
THTE	REAL	Equivalent potential temp in Kelvin
PRES	REAL	Lifted pressure

Output parameters:

PR_LTMP	REAL	Lifted temperature in Celsius

19.35 PR_M100 - MULTIPLY BY 100

This function multiplies a value by 100.

PR_M100 (VALUE)

Input parameters:

VALUE	REAL	Value

Output parameters:

PR_M100	REAL	Value * 100

19.36 PR_MHGT - COMPUTE MHGT

This function computes the moist hydrostatic height at pressure, PT, from a lower height and pressure, and the scale height in the layer. PR_SCLH can be used to compute the scale height. MHGT is computed as an integrated quantity. Thus, the lower height should have been integrated from the surface.

PR_MHGT = HB + SCALE * ALOG (PB / PT)

PR_MHGT (HB, PB, PT, SCALE)

Input parameters:

HB	REAL	Bottom height
PB	REAL	Bottom pressure
PT	REAL	Top pressure
SCALE	REAL	Scale height

Output parameters:

PR_MHGT	REAL	Moist hydrostatic height

19.37 PR_MIXR - MIXR FROM DWPC, PRES

This function computes MIXR from DWPC and PRES. The following equation is used:

```
MIXR = .62197 * ( e / ( PRES - e ) ) * 1000.

        e    =  VAPR * corr
        corr =  (1.001 + ( ( PRES - 100. ) / 900. ) * .0034)
```

University of Wisconsin green sheet.

This function can also be used for the following computations:

MIXS from TMPC and PRES
SMXR from DWPC and PALT
SMXS from TMPC and PALT

PR_MIXR (DWPC, PRES)

Input parameters:

DWPC	REAL	Dewpoint in Celsius
PRES	REAL	Pressure in millibars

Output parameters:

PR_MIXR	REAL	Mixing ratio in g/kg

19.38 PR_MMIN - INCHES/100 FROM MM

This function converts millimeters to hundredths of inches. The following equation is used:

MMIN = 3.93701 * XMILM

PR_MMIN (XMILM)

Input parameters:

XMILM	REAL	Millimeters

Output parameters:

PR_MMIN	REAL	Hundredths of inches

19.39 PR_MSKN - SKNT FROM SPED

This function computes SKNT from SPED. The following equation is used:

SKNT = SPED * 1.9438

PR_MSKN (SPED)

Input parameters:

SPED	REAL	Speed in meters/second

Output parameters:

PR_MSKN	REAL	Speed in knots

19.40 PR_NSYM - WWMO FROM WNUM

This function converts the GEMPAK weather code WNUM to the WMO weather code, WWMO, which is used to plot weather symbols.

PR_NSYM (WNUM)

Input parameters:

WNUM	REAL	GEMPAK numeric code

Output parameters:

PR_NSYM	REAL	Weather symbol number

19.41 PR_PALT - PALT FROM ALTM, SELV

This function computes PALT from ALTM and SELV. The following equation is used:

```
    PALT = ALTM * ( 1 - ( SELK * GAMUSD / To ) ) ** expo

         SELK  =  SELV / 1000
           To  =  US Std. Atmos. sea level temp in Kelvin
               =  288.
         expo  =  GRAVTY / ( GAMUSD * RDGAS ) * 1000
```

Wallace and Hobbs.

PR_PALT (ALTM, SELV)

Input parameters:

ALTM	REAL	Altimeter in millibars
SELV	REAL	Station elevation in meters

Output parameters:

PR_PALT	REAL	Pressure in millibars

19.42 PR_PKDD - DRCT FROM PSPD

This function computes DRCT from PSPD. PSPD is in the form DDFFF where DD is the wind direction in tens of degrees, and FFF is either the wind speed or wind speed plus 500, depending on the units digit of direction rounded to the nearest 5 degrees. The following equation is used:

DRCT = 5. * INT (PSPD / 500.)

PR_PKDD (PSPD)

Input parameters:

PSPD	REAL	Packed speed and direction

Output parameters:

PR_PKDD	REAL	Wind direction in degrees

19.43 PR_PKSS - SPED FROM PSPD

This function computes SPED from PSPD. PSPD is in the form DDFFF, where DD is the wind direction in tens of degrees, and FFF is either the wind speed or the wind speed plus 500, depending on the unit digit of direction rounded to the nearest 5 degrees. The following equation is used:

SPED = MOD (INT (PSPD) , 500)

PR_PKSS (PSPD)

Input parameters:

PSPD	REAL	Packed speed and direction

Output parameters:

PR_PKSS	REAL	Wind speed in knots

19.44 PR_PLCL - PLCL FROM TMPC, PRES, TLCL

This function computes PLCL from TMPC, PRES, and TLCL for an air parcel. TMPC and PRES refer to the parcel before lifting, while TLCL is the temperature at the LCL. TLCL may be computed using PR_TLCL. The equation used is a modified Poisson equation:

PLCL = PRES * (TLCL / TMPK) ** (1 / RKAPPA)

PR_PLCL (TMPC, PRES, TLCL)

Input parameters:

TMPC	REAL	Temperature in Celsius
PRES	REAL	Pressure in millibars
TLCL	REAL	LCL temperature in Kelvin

Output parameters:

PR_PLCL	REAL	LCL pressure in millibars

19.45 PR_PMSL - PMSL FROM PRES, TMPC, DWPC, SELV

This function computes PMSL from PRES, TMPC, DWPC, and SELV. The following equation is used:

```
PMSL = PRES * EXP ( ( GRAVTY * SELV ) / ( RDGAS * TVAVE ) )

        TVAVE = avg virtual temp between station and sea level
              = TVRK + ( DELTV / 2 )
        DELTV = GAMUSD * SELV / 1000
```

Wallace and Hobbs.

```
PR_PMSL ( PRES, TMPC, DWPC, SELV )
```

Input parameters:

PRES	REAL	Station pressure in millibars
TMPC	REAL	Temperature in Celsius
DWPC	REAL	Dewpoint in Celsius
SELV	REAL	Station elevation in meters

Output parameters:

PR_PMSL	REAL	Mean sea level pressure in mb

19.46 PR_PRES - PRES FROM TMPC, THTA

This function computes PRES from TMPC and THTA. Poisson's equation is used:

```
PRES = 1000. * ( PR_TMCK (TMPC) / THTA ) ** (1 / RKAPPA)
```

PR_PRES (TMPC, THTA)

Input parameters:

TMPC	REAL	Temperature in Celsius
THTA	REAL	Potential temperature in Kelvin

Output parameters:

PR_PRES	REAL	Station pressure in millibars

19.47 PR_PSPD - PSPD FROM DRCT, SPED

This function computes PSPD from DRCT and SPED. PSPD is in the form DDFFF, where DD is the wind direction in tens of degrees, and FFF is either the wind speed or wind speed plus 500, depending on the unit digit of direction rounded to the nearest 5 degrees. The following equation is used:

```
PSPD  =  JDRCT * 500 + JSPED
         JDRCT = NINT ( DRCT / 5 )
         JSPED = NINT ( SPED )
```

PR_PSPD (DRCT, SPED)

Input parameters:

DRCT	REAL	Wind direction in degrees
SPED	REAL	Wind speed

Output parameters:

PR_PSPD	REAL	Packed speed and direction

19.48 PR_RELH - RELH FROM TMPC, DWPC

This function computes RELH from TMPC and DWPC. The following equation is used:

```
RELH  =  VAPR / VAPS * 100

         VAPR = vapor pressure
              = PR_VAPR ( DWPC )
         VAPS = saturation vapor pressure
              = PR_VAPR ( TMPC )
```

PR_RELH (TMPC, DWPC)

Input parameters:

TMPC	REAL	Temperature in Celsius
DWPC	REAL	Dewpoint in Celsius

Output parameters:

PR_RELH	REAL	Relative humidity in percent

19.49 PR_RHDP - DWPC FROM TMPC, RELH

This function computes DWPC from TMPC and RELH. The following equation is used:

```
DWPC = 243.5 * LN (6.112) - 243.5 * LN (VAPR) /
       ( LN (VAPR) - LN (6.112) - 17.67 )

       VAPR = VAPS * RELH
       VAPS = saturation vapor pressure
            = PR_VAPR ( TMPC )
```

Note: If DWPC is less than -190 degrees C, it is treated as missing data.

PR_RHDP (TMPC, RELH)

Input parameters:

TMPC	REAL	Temperature in Celsius
RELH	REAL	Relative humidity in percent

Output parameters:

PR_RHDP	REAL	Dewpoint in Celsius

19.50 PR_RWSH - RWSH FROM ASHEVILLE CODE

This function computes RWSH from INUM, an 8-integer array. INUM, the original 8-digit Asheville integer code is converted to RWSH, a 7-digit real number which can be stored in a surface file. Note that real numbers have only 7-digit precision. Meaningful data in column 1 is added to column 8. Thus, some data that were originally in column 8 may be lost. The data that can be lost in column 8 include 1) smoke, 2) haze, 3) smoke-and-haze, 4) dust, and 5) blowing snow. A packed real number is then constructed from columns 2 through 8 as 2345.678.

PR_RWSH (INUM)

Input parameters:

INUM (8)	INTEGER	NCDC 8-digit weather code

Output parameters:

PR_RWSH	REAL	Packed 7-digit weather code

19.51 PR_SALI - SALI FROM ALTI

This function computes SALI from ALTI. SALI is an abbreviated altimeter code in inches which contains the unit digit and the first two digits after the decimal points. ALTI is multiplied by 100 truncated, and the original tens digit dropped. The following equation is used:

SALI = NINT (MOD (ALTI, 10) * 100)

PR_SALI (ALTI)

Input parameters:

ALTI	REAL	Altimeter setting in inches

Output parameters:

PR_SALI	REAL	Abbreviated standard altimeter

19.52 PR_SCLH - COMPUTE SCALE HEIGHT

This function computes SCLH from TB, TT, TDB, TDT, PB, and PT. SCLH, the scale height in a layer, can then be used to compute the moist hydrostatic height. The following equation is used:

```
SCLH  =  ( RDGAS / GRAVTY ) * TAV

         TAV    = average virtual temperature in layer
                = ( TVIRTB + TVIRTT ) / 2
         TVIRTB = virtual temperature at bottom
         TVIRTT = virtual temperature at top
```

PR_SCLH (TB, TT, TDB, TDT, PB, PT)

Input parameters:

TB	REAL	Bottom temperature in Celsius
TT	REAL	Top temperature in Celsius
TDB	REAL	Bottom dewpoint in Celsius
TDT	REAL	Top dewpoint in Celsius
PB	REAL	Bottom pressure in millibars
PT	REAL	Top pressure in millibars

Output parameters:

PR_SCLH	REAL	Scale height in meters

19.53 PR_SPED - SPED FROM UWND, VWND

This function computes SPED from UWND and VWND. The following equation is used:

SPED = SQRT ((UWND**2) + (VWND**2))

This function computes SKNT if UKNT and VKNT are input.

PR_SPED (UWND, VWND)

Input parameters:

UWND	REAL	U component of velocity
VWND	REAL	V component of velocity

Output parameters:

PR_SPED	REAL	Wind speed

19.54 PR_STDZ - STDZ FROM PRES, HGHT

This function computes a standard height used on upper-air charts. For data below 500 mb, the standard height is the last three digits of the height. For data at and above 500 mb, the height is the last three digits of the height in decameters.

PR_STDZ (PRES, HGHT)

Input parameters:

PRES	REAL	Pressure in millibars
HGHT	REAL	Height in meters

Output parameters:

PR_STDZ	REAL	Abbreviated height

19.55 PR_THTA - THTA FROM TMPC, PRES

This function computes THTA from TMPC and PRES using Poisson's equation:

```
THTA = TMPK * ( 1000 / PRES ) ** RKAPPA
```

This function also computes STHA from TMPC and PALT, THTV from TVRC and PRES, and THTV from TVRC and PALT.

PR_THTA (TMPC, PRES)

Input parameters:

TMPC	REAL	Temperature in Celsius
PRES	REAL	Pressure in millibars

Output parameters:

PR_THTA	REAL	Potential temperature in K

19.56 PR_THTE - THTE FROM PRES, TMPC, DWPC

This function computes THTE from PRES, TMPC, DWPC. In the calculation, MIXR depends on PRES and DWPC; TLCL depends on TMPC and DWPC. The following equation is used:

```
          THTE = THTAM * EXP [ ( 3.376/TLCL - .00254 ) *
                               ( MIXR * ( 1 + .81*.001*MIXR ) ) ]

               THTAM = potential temperature of moist air
                     = TMPK * (1000 / PRES) ** E
                   E = RKAPPA * ( 1 - ( .28 * .001 * MIXR ) )
```

Bolton.

PR_THTE (PRES, TMPC, DWPC)

Input parameters:

PRES	REAL	Pressure in millibars
TMPC	REAL	Temperature in Celsius
DWPC	REAL	Dewpoint in Celsius

Output parameters:

PR_THTE	REAL	Equivalent potential temp in K

19.57 PR_TLCL - TLCL FROM TMPC, DWPC

This function computes temperature at the Lifted Condensation Level for a parcel of air given TMPC and DWPC. The following equation is used:

TLCL = [1 / (1 / (DWPK-56) + ALOG (TMPK/DWPK) / 800)] + 56

Bolton.

PR_TLCL (TMPC, DWPC)

Input parameters:

TMPC	REAL	Temperature in Celsius
DWPC	REAL	Dewpoint in Celsius

Output parameters:

PR_TLCL	REAL	LCL temperature in Kelvin

19.58 PR_TMCF - TMPF FROM TMPC

This function computes TMPF from TMPC. The following equation is used:

$$TMPF = (TMPC * 9 / 5) + 32$$

PR_TMCF (TMPC)

Input parameters:

TMPC	REAL	Temperature in Celsius

Output parameters:

PR_TMCF	REAL	Temperature in Fahrenheit

19.59 PR_TMCK - TMPK FROM TMPC

This function computes TMPK from TMPC. The following equation is used:

$$TMPK = TMPC + 273.16$$

PR_TMCK (TMPC)

Input parameters:

TMPC	REAL	Temperature in Celsius

Output parameters:

PR_TMCK	REAL	Temperature in Kelvin

19.60 PR_TMFC - TMPC FROM TMPF

This function computes TMPC from TMPF. The following equation is used:

$$TMPC = (TMPF - 32) * 5 / 9$$

PR_TMFC (TMPF)

Input parameters:

TMPF	REAL	Temperature in Fahrenheit

Output parameters:

PR_TMFC	REAL	Temperature in Celsius

19.61 PR_TMFK - TMPK FROM TMPF

This function computes TMPK from TMPF. The following equation is used:

TMPK = (TMPF - 32) * 5 / 9 + 273.16

PR_TMFK (TMPF)

Input parameters:		
TMPF	REAL	Temperature in Fahrenheit

Output parameters:		
PR_TMFK	REAL	Temperature in Kelvin

19.62 PR_TMKC - TMPC FROM TMPK

This function computes TMPC from TMPK. The following equation is used:

$$TMPC = TMPK - 273.16$$

PR_TMKC (TMPK)

Input parameters:

TMPK	REAL	Temperature in Kelvin

Output parameters:

PR_TMKC	REAL	Temperature in Celsius

19.63 PR_TMKF - TMPF FROM TMPK

This function computes TMPF from TMPK. The following equation is used:

$$TMPF = ((TMPK - 273.16) * 9 / 5) + 32$$

PR_TMKF (TMPK)

Input parameters:

TMPK	REAL	Temperature in Kelvin

Output parameters:

PR_TMKF	REAL	Temperature in Fahrenheit

19.64 PR_TMPK - TMPK FROM PRES, THTA

This function computes TMPK from PRES and THTA. The Poisson equation is used:

TMPK = THTA * (PRES / 1000) ** RKAPPA

PR_TMPK (PRES, THTA)

Input parameters:

PRES	REAL	Pressure in millibars
THTA	REAL	Potential temperature in K

Output parameters:

PR_TMPK	REAL	Temperature in Kelvin

19.65 PR_TMST - PARCEL TEMPERATURE

This function computes TMST from THTE, PRES, TGUESS. TMST is the parcel temperature at level PRES on a specified moist adiabat (THTE). The computation is an iterative Newton-Raphson technique of the form:

```
x = x(guess) + [ f( x ) - f( x(guess) ) ] / f'( x(guess) )

   f' is approximated with finite differences
   f' = [ f( x(guess) + 1 ) - f( x(guess) ) ] / 1
```

If TGUESS is 0, a reasonable first guess will be made.

Convergence is not guaranteed for extreme input values. If the computation does not converge after 100 iterations, the missing data value will be returned.

PR_TMST (THTE, PRES, TGUESS)

Input parameters:

THTE	REAL	Equivalent potential temp in K
PRES	REAL	Pressure in millibars
TGUESS	REAL	First guess temperature in K

Output parameters:

PR_TMST	REAL	Parcel temperature in Kelvin

19.66 PR_TVRK - TVRK FROM TMPC, DWPC, PRES

This function computes TVRK from TMPC, DWPC and PRES, where DWPC and PRES are used to compute MIXR. The following equation is used:

TVRK = TMPK * (1 + .001 * MIXR / .62197) / (1 + .001 * MIXR)

If DWPC is missing, dry air is assumed and TMPK is returned.

PR_TVRK (TMPC, DWPC, PRES)

Input parameters:

TMPC	REAL	Temperature in Celsius
DWPC	REAL	Dewpoint in Celsius
PRES	REAL	Pressure in millibars

Output parameters:

PR_TVRK	REAL	Virtual temperature in Kelvin

19.67 PR_UWND - UWND FROM SPED, DRCT

This function computes UWND from SPED and DRCT or UKNT from SKNT and DRCT. The following equation is used:

$$UWND = -SIN (DRCT) * SPED$$

PR_UWND (SPED, DRCT)

Input parameters:

SPED	REAL	Wind speed
DRCT	REAL	Wind direction in degrees

Output parameters:

PR_UWND	REAL	U component

19.68 PR_VAPR - VAPR FROM DWPC

This function computes VAPR from DWPC. The following equation is used:

VAPR = 6.112 * EXP [(17.67 * DWPC) / (DWPC + 243.5)]

Bolton.

This function will compute VAPS if TMPC is input.

PR_VAPR (DWPC)

Input parameters:

DWPC	REAL	Dewpoint in Celsius

Output parameters:

PR_VAPR	REAL	Vapor pressure in millibars

19.69 PR_VWND - VWND FROM SPED, DRCT

This function computes VWND from SPED and DRCT or VKNT from SKNT and DRCT. The following equation is used:

VWND = -COS (DRCT) * SPED

PR_VWND (SPED, DRCT)

Input parameters:		
SPED	REAL	Wind speed
DRCT	REAL	Wind direction in degrees

Output parameters:		
PR_VWND	REAL	V component

19.70 PR_WCMP - WIND COMPONENT

This function computes the wind component toward a specified direction. The following equation is used:

WCMP = -COS (DRCT - DCMP) * SPED

WCMP = component of wind in meters/second
DRCT = wind direction in degrees
DCMP = direction of desired component
SPED = wind speed in meters/second

PR_WCMP (DRCT, SPED, DCMP)

Input parameters:

SPED	REAL	Wind speed in meters/second
DRCT	REAL	Wind direction in degrees

19.71 PR_WNML - NORMAL WIND COMPONENT

This function computes the wind component toward a direction 90 degrees counterclockwise of a specified direction. If no direction is specified, the component toward north is returned. The following equation is used:

WNML = -COS (DRCT - (DCMP-90)) * SPED

WNML = component of wind in meters/second
DRCT = wind direction in degrees
DCMP = specified direction
SPED = wind speed in meters/second

PR_WNML (DRCT, SPED, DCMP)

Input parameters:

DRCT	REAL	Wind direction in degrees
SPED	REAL	Wind speed in meters/sec
DCMP	REAL	Input direction in degrees

19.72 PR_ZALT - ZALT FROM ALTM, PRES

This function computes a height from ALTM and PRES. This function is used to estimate height at various pressure levels from the altimeter in millibars. The PC library computes ZMSL, Z000, Z950, Z850, Z800 by calling PR_ZALT with PRES equal to PSML, 1000, 950, 850 and 800 respectively. The following equation is used:

```
ZALT  =  [ To * ( 1 - ( ( PRES/ALTM ) ** expo ) ] / GAMMA

      GAMMA  = GAMUSD / 1000
         To  =  US Std. Atmos. sea level temp in Kelvin
             =  288.
       expo  =  ( GAMMA * RDGAS ) / GRAVTY
```

PR_ZALT (ALTM, PRES)

Input parameters:

ALTM	REAL	Altimeter in millibars
PRES	REAL	Pressure in millibars

Output parameters:

PR_ZALT	REAL	Height in meters

CHAPTER 20

CHARACTER TRANSLATION (PT) LIBRARY

PT_CCNM	Compute numeric cloud coverage
PT_CLDN	Compute character cloud coverage
PT_CLDS	Compute cloud cover for 3 levels
PT_CMCL	Compute combined height and cover
PT_OCHR	Compute ozone character code
PT_PWTH	Compute character past weather
PT_SALT	Compute 3-character pressure code
PT_WASH	Compute Asheville weather code
PT_WCOD	Compute character weather code
PT_WNUM	Compute numeric weather code
PT_WSYM	Compute weather symbol code
PT_WTHR	Compute old character weather
PT_WTMO	Compute WMO character weather
PT_W604	Compute old numeric weather

Character Translation (PT) Library Summary

The character translation library contains functions to convert numeric codes into character strings and vice versa. The functions which output character data are called by the GEMPAK parameter conversion (PC) library. Since all data in GEMPAK files must be stored as real values, functions to translate character data to numeric codes are included for use in data ingest programs. When creating a GEMPAK file, the numeric names MUST be used.

Several methods for storing surface weather reports are available. The parameter names and conversion functions are included in the following table. For new files, WNUM should be used.

Data Type	Char Name	Num Name	N-->C Conv	C-->N Conv
general weather	WCOD	WNUM	PT_WCOD	PT_WNUM
old weather	WTHR	W604	PT_WTHR	PT_W604
WMO weather	WTMO	WWMO	PT_WTMO	
Asheville	WASH	RWSH	PT_WASH	
WMO past weather	PWTH	PWWM	PT_PWTH	

CHARACTER TRANSLATION (PT) LIBRARY

PT Library Calls

PT_CCNM	(xcld)
PT_CLDN	(clcx)
PT_CLDS	(cmbc)
PT_CMCL	(comx)
PT_OCHR	(ozone)
PT_PWTH	(pwwm)
PT_SALT	(rval)
PT_WASH	(rwsh)
PT_WCOD	(wnum)
PT_WNUM	(wthr)
PT_WSYM	(wthr)
PT_WTHR	(w604)
PT_WTMO	(wwmo)
PT_W604	(wthr)

20.1 PT_CCNM - COMPUTE NUMERIC CLOUD COVERAGE

This function translates character cloud coverage into a numeric cloud coverage:

CLCx = PT_CCNM (xCLD)

' '	= 0	X	= 5
CLR	= 1	-SCT	= 6
SCT	= 2	-BKN	= 7
BKN	= 3	-OVC	= 8
OVC	= 4	-X	= 9

The characters must be left-justified in the string.

PT_CCNM (XCLD)

Input parameters:

XCLD	CHAR*	Character cloud coverage

Output parameters:

PT_CCNM	REAL	Numeric cloud code

20.2 PT_CLDN - COMPUTE CHARACTER CLOUD COVERAGE

This character function translates numeric cloud coverage into character cloud coverage:

xCLD = PT_CLDN (CLCx)

0 = ' '	5 = X
1 = CLR	6 = -SCT
2 = SCT	7 = -BKN
3 = BKN	8 = -OVC
4 = OVC	9 = -X

The characters are left-justified in the string.

PT_CLDN (CLCX)

Input parameters:

CLCX	REAL	Numeric cloud code

Output parameters:

PT_CLDN	CHAR*	Character cloud coverage

20.3 PT_CLDS - COMPUTE CLOUD COVER FOR 3 LEVELS

This character function converts packed three-level numeric cloud coverage into packed three-level character cloud coverage:

CLDS = PT_CLDS (CMBC)

The input parameter may be computed using PR_CMBC. The individual cloud conversions are:

0 = _ (underscore)
1 = C
2 = S
3 = B
4 = O
5 = X
6 = -S
7 = -B
8 = -O
9 = -X

EXAMPLE: CMBC = 263.
PT_CLDS = S-SB

The characters are left-justified in the output string.

PT_CLDS (CMBC)

Input parameters:

CMBC	REAL	Combined cloud coverage

Output parameters:

PT_CLDS	CHAR*	Char combined cloud coverage

20.4 PT_CMCL - COMPUTE COMBINED HEIGHT AND COVER

This character function returns the character value for the combined cloud height and cloud coverage:

CLDx = PT_CMCL (COMx)

The input value COMx may be computed from the cloud height and coverage using the function PR_COMX. The output height is given in hundreds of feet; the cloud cover code is the short code:

```
0 ---> _ (underscore)
1 ---> C
2 ---> S
3 ---> B
4 ---> O
5 ---> X
6 ---> -S
7 ---> -B
8 ---> -O
9 ---> -X
```

```
Example:  COMX    = 1507.
          PT_CMCL = 150-B
```

The characters are left justified in the string.

PT_CMCL (COMX)

Input parameters:

COMX	REAL	Combined height & coverage

Output parameters:

PT_CMCL	CHAR*	Character height & coverage

20.5 PT_OCHR - COMPUTE OZONE CHARACTER CODE

This character function converts a numeric ozone value into a character code. The intervals cover 5 units:

0 - 4	a
5 - 9	A
10 - 14	b
.	.
.	.
.	.

PT_OCHR (OZONE)

Input parameters:

OZONE	REAL	Ozone value

Output parameters:

PT_OCHR	CHAR*	Character code

20.6 PT_PWTH - COMPUTE CHARACTER PAST WEATHER

This character function converts a numeric WMO past weather code, PWWM, into a character weather code:

PWTH = PT_PWTH (PWWM)

The values for the numeric values are:

0 = Cloud covering less than 1/2 sky
1 = Cloud covering more than 1/2 sky during part of period
    and less than 1/2 during part of period
2 = Cloud covering more than 1/2 sky
3 = Sandstorm, dust storm or blowing snow
4 = Fog, ice fog, thick haze or thick smoke
5 = Drizzle
6 = Rain
7 = Snow, mixed rain and snow, or ice pellets
8 = Showers
9 = Thunderstorm with or without precipitation

The conversion is:

0 = ' '	5 = L
1 = ' '	6 = R
2 = ' '	7 = S
3 = BD	8 = RW
4 = F	9 = T

PT_PWTH (PWWM)

Input parameters:

PWWM	REAL	Numeric past weather code

Output parameters:

PT_PWTH	CHAR*	Character past weather

20.7 PT_SALT - COMPUTE 3-CHARACTER PRESSURE CODE

This function takes a real number and converts the integral part into a 3-character string. Leading blanks are changed to 0. It can be used to output abbreviated pressure and altimeter values.

PT_SALT (RVAL)

Input parameters:

RVAL	REAL	Value

Output parameters:

PT_SALT	CHAR*	Three-character pressure code

20.8 PT_WASH - COMPUTE ASHEVILLE WEATHER CODE

This character function converts a real number which represents an Asheville numeric weather code into a character string:

WASH = PT_WASH (RWSH)

The numeric code is stored as 1234.567 where the conversions for the columns are:

Column 1:	Column 2:	Column 3:	Column 4:
0 = None	0 = None	0 = None	0 = None
1 = R-	1 = None	1 = S-	1 = SW-
2 = R	2 = None	2 = S	2 = SW
3 = R+	3 = None	3 = S+	3 = SW+
4 = RW-	4 = L-	4 = SP-	4 = None
5 = RW	5 = L	5 = SP	5 = None
6 = RW+	6 = L+	6 = SP+	6 = None
7 = ZR-	7 = ZL-	7 = none	7 = SG-
8 = ZR	8 = ZL	8 = IC	8 = SG
9 = ZR+	9 = ZL+		9 = SG+

Column 5:	Column 6:	Column 7:
0 = None	0 = None	0 = None
1 = IP-	1 = F	1 = K
2 = IP	2 = IF	2 = H
3 = IP+	3 = GF	3 = KH
4 = None	4 = BD	4 = D
5 = A	5 = BN	5 = BS
6 = None		6 = T
7 = None		7 = T+
8 = AP		8 = TOR
		9 = Q

NOTE: the original scheme had a sixth value in col 7 for blowing spray, which has been omitted. The weather codes for 6 through 9 in column 7 were added to this column from the original column 1.

PT_WASH (RWSH)

Input parameters:

RWSH	REAL	Asheville numeric code

Output parameters:

PT_WASH	CHAR*	Character weather code

20.9 PT_WCOD - COMPUTE CHARACTER WEATHER CODE

This function converts a GEMPAK numeric weather code, WNUM, into a character code, WCOD:

WCOD = PT_WCOD (WNUM)

WCOD can be converted to WNUM using PT_WNUM.

PT_WCOD (WNUM)

Input parameters:

WNUM	REAL	Weather number

Output parameters:

PT_WCOD	CHAR*	Character weather string

20.10 PT_WNUM - COMPUTE NUMERIC WEATHER CODE

This function converts any character weather code into a GEMPAK weather number, WNUM:

WNUM = PT_WNUM (WTHR)

WNUM can be converted to a character weather code, WCOD, using the function PT_WCOD. The range of numbers which might result is -3 to 512000.

PT_WNUM (WTHR)

Input parameters:

WTHR	CHAR*	Character weather string

Output parameters:

PT_WNUM	REAL	Weather number

20.11 PT_WSYM - COMPUTE WEATHER SYMBOL CODE

This function converts a character weather code, WTHR, into a synoptic numeric code for the weather symbol number, which is used to draw weather symbols.

WSYM = PT_WSYM (WTHR)

The conversion that is used is based upon that which the National Meteorological Center (NMC) uses to convert hourly alphanumeric characters to the synoptic weather code. This conversion is shown in the following table. Note that some GEMPAK codes have been added. These are denoted in lower case. Also note, 10 has been added to codes that have two symbols for the snow case as a convention.

```
  4 = K
  5 = KH HK H
  6 = KD HD d
  7 = BD BN N BY
 10 = KF HF F IF
 12 = GF HGF
 17 = T+ T
 18 = q
 19 = TORNA FUNNE WATER
 34 = bd+
 38 = BS
 39 = bs+
 51 = L-
 53 = L LS
 55 = L+
 56 = ZL- ZLW-
 57 = ZL+ ZL
 58 = R-L- L-R-
 59 = RL L+R+ LR
 61 = R-
 63 = R
 65 = R+
 66 = ZR- ZRW-
 67 = ZR+ ZR
 68 = L-S- R-S- r+s- r-s rs-
 69 = L+S+ R+S+ RS r-s+
 71 = S-
 73 = S
 75 = S+
 76 = IC
 77 = SGW SG SGW-
 79 = IP R-IP
 80 = RW-
 81 = RW+ RW
 83 = RW - S
 84 = RWSW RW+S
 85 = SW-
 86 = SW+ SW
 87 = AP- IPW- SPW- SP-
 88 = AP+ SPW AP IPW IPW+ SP
 89 = RW-A A-
 90 = RW+A A A+ RWA
 95 = TRW RT TR TRW- T-R
 96 = T-A TA trwa trw-a
 97 = TRW+ T+R tr+
 98 = TBN TBD T+BN T+BD tD
 99 = T+A trw+a
105 = T-S TSW TSW- TS
107 = TSW+ T+RS T+S
```

PT_WSYM (WTHR)

Input parameters:

WTHR	CHAR*	Character weather code

Output parameters:

PT_WSYM	REAL	Numeric weather code

20.12 PT_WTHR - COMPUTE OLD CHARACTER WEATHER

This character function converts a numeric weather code, W604, into a character string, WTHR:

WTHR = PT_WTHR (W604)

W604 was a numeric weather code used with an old 604 data ingest system in which 64 unique weather reports were recognized.

The conversion is:

0 = ' '	22 = L+	44 = R+S
1 = T	23 = SP	45 = RS-
2 = R	24 = SG	46 = RS+
3 = S	25 = BY	47 = ZL-
4 = F	26 = BN	48 = ZL+
5 = H	27 = BD	49 = TSW
6 = K	28 = IF	50 = TSW-
7 = L	29 = RS	51 = TSW+
8 = R+	30 = TRW	52 = TRW-
9 = R-	31 = TR-	53 = TRW+
10 = S+	32 = TR+	54 = TRWA
11 = S-	33 = TS-	55 = R-S-
12 = RW	34 = TS+	56 = R-S+
13 = SW	35 = ZR-	57 = R+S-
14 = TR	36 = ZR+	58 = R+S+
15 = TS	37 = SW-	59 = TRW+A
16 = ZL	38 = SW+	60 = TRW-A
17 = ZR	39 = SG-	61 = TORNA
18 = IP	40 = SG+	62 = FUNNE
19 = GF	41 = RW-	63 = WATER
20 = BS	42 = RW+	
21 = L-	43 = R-S	

WTHR can be converted to W604 using the function PT_W604.

PT_WTHR (W604)

Input parameters:

W604	REAL	Numeric weather code

Output parameters:

PT_WTHR	CHAR*	Character weather code

20.13 PT_WTMO - COMPUTE WMO CHARACTER WEATHER

This character function converts a numeric WMO weather code, WWMO, into a character code:

WTMO = PT_WTMO (WWMO)

The conversion is:

0 =		34 =	BD+	67 =	ZR
1 =		35 =	BD+	68 =	R-S-
2 =		36 =	BS	69 =	RS
3 =		37 =	BS+	70 =	S-
4 =	K	38 =	BS	71 =	S-
5 =	H	39 =	BS+	72 =	S
6 =	D	40 =		73 =	S
7 =	BD	41=	F	74 =	S+
8 =	BD	42 =	F	75 =	S+
9 =	BD	43 =	F	76 =	IN
10 =	F-	44 =	F	77 =	SG
11 =	GF	45 =	F	78 =	S-
12 =	GF	46 =	F	79 =	IP
13 =		47 =	F	80 =	RW-
14 =		48 =	F	81 =	RW
15 =		49 =	F	82 =	RW+
16 =		50 =	L-	83 =	RW-SW-
17 =	T	51 =	L-	84 =	RWSW
18 =	Q	52 =	L-	85 =	SW-
19 =	FUNNE	53 =	L	86 =	SW
20 =		54 =	L+	87 =	SP-
21 =		55 =	L+	88 =	SP
22 =		56 =	ZL-	89 =	A-
23 =		57 =	ZL	90 =	A
24 =		58 =	R-L-	91 =	R-
25 =		59 =	RL	92 =	R
26 =		60 =	R-	93 =	RS
27 =		61 =	R-	94 =	R+S+
28 =		62 =	R	95 =	TRW-
29 =		63 =	R	96 =	TRW-A
30 =	BD	64 =	R+	97 =	TRW+
31 =	BD	65 =	R+	98 =	TD
32 =	BD	66 =	ZR-	99 =	TRW+A
33 =	BD+				

PT_WTMO (WWMO)

Input parameters:

WWMO	REAL	Numeric weather code

Output parameters:

PT_WTMO	CHAR*	Character weather code

20.14 PT_W604 - COMPUTE OLD NUMERIC WEATHER

This function converts a character weather code, WTHR, into a numeric code, W604:

W604 = PT_W604 (WTHR)

The conversion is:

0 =	' '	22 =	L+	44 =	R+S
1 =	T	23 =	SP	45 =	RS-
2 =	R	24 =	SG	46 =	RS+
3 =	S	25 =	BY	47 =	ZL-
4 =	F	26 =	BN	48 =	ZL+
5 =	H	27 =	BD	49 =	TSW
6 =	K	28 =	IF	50 =	TSW-
7 =	L	29 =	RS	51 =	TSW+
8 =	R+	30 =	TRW	52 =	TRW-
9 =	R-	31 =	TR-	53 =	TRW+
10 =	S+	32 =	TR+	54 =	TRWA
11 =	S-	33 =	TS-	55 =	R-S-
12 =	RW	34 =	TS+	56 =	R-S+
13 =	SW	35 =	ZR-	57 =	R+S-
14 =	TR	36 =	ZR+	58 =	R+S+
15 =	TS	37 =	SW-	59 =	TRW+A
16 =	ZL	38 =	SW+	60 =	TRW-A
17 =	ZR	39 =	SG-	61 =	TORNA
18 =	IP	40 =	SG+	62 =	FUNNE
19 =	GF	41 =	RW-	63 =	WATER
20 =	BS	42 =	RW+		
21 =	L-	43 =	R-S		

W604 can be converted to WTHR using the function PT_WTHR.

PT_W604 (WTHR)

Input parameters:

WTHR	CHAR*	Character weather code

Output parameters:

PT_WTHR	REAL	Numeric weather code

CHAPTER 21

AIRWAYS DECODER (RA) LIBRARY

RA_CHCK	Check for time and station in file
RA_CLEV	Process cloud information
RA_DECD	Decode airways report
RA_GFLD	Break report into fields
RA_GRPT	Get report from bulletin
RA_RHDR	Get header information
RA_RTIM	Get report time
RA_TMST	Set time and station

AIRWAYS DECODER (RA) LIBRARY

Surface Airways Decoder (RA) Library Summary

The RA library contains subroutines to decode and store surface airways reports.

The airways decoder must have access to individual airways reports. RA_GRPT extracts reports from bulletins.

Before the report can be decoded, the subroutine RA_GFLD must be called. This subroutine breaks the report into parts and saves them in a common area which can be accessed by the other subroutines. RA_RHDR can then be called to get the report header information. RA_DECD decodes the rest of the bulletin.

The decoder, DCSURF, is a realtime decoder which can be used as an example.

ERROR MESSAGES:

In general, the errors encountered in the RA library are not fatal to continued execution, but just flag a problem with a particular report. Thus, it seems unlikely that a programmer will want to print a message every time the return code is non-zero. However, the return code values are summarized here as an aid to program development.

[RA -1]	The time is invalid.
[RA -2]	There are no more reports.
[RA -3]	No wind group was found.
[RA -4]	The time cannot be set.
[RA -5]	The station cannot be set.
[RA -6]	The report cannot be decoded.

AIRWAYS DECODER (RA) LIBRARY

RA Library Calls

RA_CHCK (isffln, dattim, stid, / timflg, stnflg, datflg, iret)

RA_CLEV (cldtyp, cldhgt, ncld, / chc1, chc2, chc3, iret)

RA_DECD (irpntr, coun, maxcld, / cldtyp, cldhgt, ncld, vsby, wcod, wnum, pres, tmpf, dwpf, sknt, drct, gust, alti, iret)

RA_GFLD (report, lenr, / iret)

RA_GRPT (bultin, lenb, / ibpnt, report, lenr, iret)

RA_RHDR (/ irpntr, stid, rpttyp, corflg, autotp, ihour, iminut, iret)

RA_RTIM (isdtar, btime, irhour, irmin, / irdtar, rtime, iret)

RA_TMST (isffln, dattim, stid, addstn, cirflg, / datflg, iret)

21.1 RA_CHCK - CHECK FOR TIME AND STATION IN FILE

This subroutine checks to see if a time and station are already in the surface file. If the time and/or station is not found, the logical variables TIMFLG and/or STNFLG is set to false. If both the time and station are found, datflg will be true if data for this station has already been added to the file.

RA_CHCK (ISFFLN, DATTIM, STID, TIMFLG, STNFLG, DATFLG, IRET)

Input parameters:

ISFFLN	INTEGER	Sounding file number
DATTIM	CHAR*15	Nominal date/time
STID	CHAR*	Station identifier

Output parameters:

TIMFLG	LOGICAL	Time found flag
STNFLG	LOGICAL	Station found flag
DATFLG	LOGICAL	Data already in file flag
IRET	INTEGER	Return code 0 = normal return

21.2 RA_CLEV - PROCESS CLOUD INFORMATION

This subroutine uses the cloud information decoded from an airways report and returns it encoded in three combined cloud height and coverage reports. If -X (partially obscured) is reported, 1000 is added to the first report. The combined value is the height * 10 + coverage.

RA_CLEV (CLDTYP, CLDHGT, NCLD, CHC1, CHC2, CHC3, IRET)

Input parameters:

CLDTYP (NCLD)	REAL	GEMPAK cloud types
CLDHGT (NCLD)	REAL	Cloud height in hundreds of feet
NCLD	INTEGER	Number of cloud reports

Output parameters:

CHC1	REAL	Cloud report 1
CHC2	REAL	Cloud report 2
CHC3	REAL	Cloud report 3
IRET	INTEGER	Return code 0 = normal return

21.3 RA_DECD - DECODE AIRWAYS REPORT

This subroutine decodes a surface airways report. RA_GFLD must be called before this subroutine is called. IRPNTR must point to the first field after the header.

```
RA_DECD  ( IRPNTR, COUN, MAXCLD, CLDTYP, CLDHGT, NCLD, VSBY,
           WCOD, WNUM, PRES, TMPF, DWPF, SKNT, DRCT, GUST, ALTI,
           IRET )
```

Input parameters:

IRPNTR	INTEGER	First field after header
COUN	CHAR*	Country name
MAXCLD	INTEGER	Maximum number of clouds

Output parameters:

CLDTYP (NCLD)	REAL	GEMPAK cloud numeric types
CLDHGT (NCLD)	REAL	Cloud heights
NCLD	INTEGER	Number of cloud reports
VSBY	REAL	Visibility in miles
WCOD	CHAR*	GEMPAK weather code
WNUM	REAL	GEMPAK weather number
PRES	REAL	Pressure in millibars
TMPF	REAL	Temperature in F
DWPF	REAL	Dewpoint temp in F
SKNT	REAL	Wind speed in knots
DRCT	REAL	Wind direction in degrees
GUST	REAL	Wind gusts in knots
ALTI	REAL	Altimeter in inches
IRET	INTEGER	Return code 0 = normal return -6 = report not decoded

21.4 RA_GFLD - BREAK REPORT INTO FIELDS

This subroutine divides a surface airways report into individual fields for decoding. Fields must be separated by blanks or slashes. Numbers and non-numeric strings are stored in separate fields. A slash is considered a separate field. Unprintable characters must be replaced by blanks before this subroutine is called. The fields are stored in / RACMN /.

RA_GFLD (REPORT, LENR, IRET)

Input parameters:

REPORT	CHAR*	AIRWAYS report
LENR	INTEGER	Length of report

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = invalid report

21.5 RA_GRPT - GET REPORT FROM BULLETIN

This subroutine gets the next report from a surface bulletin. Reports must begin with the control character, RS (= 30).

RA_GRPT (BULTIN, LENB, IBPNT, REPORT, LENR, IRET)

Input parameters:

BULTIN	CHAR*	Bulletin
LENB	INTEGER	Bulletin length

Input and Output parameters:

IBPNT	INTEGER	Pointer in bulletin

Output parameters:

REPORT	CHAR*	Report
LENR	INTEGER	Length of report
IRET	INTEGER	Return code 0 = normal return -2 = no more reports

21.6 RA_RHDR - GET HEADER INFORMATION

This subroutine gets the header information from an airways report.

RA_RHDR (IRPNTR, STID, RPTTYP, CORFLG, AUTOTP, IHOUR, IMINUT, IRET)

Output parameters:

IRPNTR	INTEGER	First field after header
STID	CHAR*	Station identifier
RPTTYP	CHAR*	Report type
CORFLG	LOGICAL	Correction flag
AUTOTP	CHAR*	Automatic station type
IHOUR	INTEGER	Hour
IMINUT	INTEGER	Minute
IRET	INTEGER	Return code 0 = normal return -2 = incomplete report

21.7 RA_RTIM - GET REPORT TIME

This subroutine combines an integer system time, the bulletin time containing the day, month and hour and the report day and hour into an observation time. It is assumed that the system time accurately reflects the year and month of the observation and is later than that time.

RA_RTIM (ISDTAR, BTIME, IRHOUR, IRMIN, IRDTAR, RTIME, IRET)

Input parameters:

ISDTAR (5)	INTEGER	System time
BTIME	CHAR*	Bulletin day, hour, minute
IRHOUR	INTEGER	Report hour
IRMIN	INTEGER	Report minute

Output parameters:

IRDTAR (5)	INTEGER	Integer report time
RTIME	CHAR*	Report date/time
IRET	INTEGER	Return code 0 = normal return -1 = invalid time

21.8 RA_TMST - SET TIME AND STATION

This subroutine sets the time and station in a surface data file. If the station has already reported, the flag DATFLG is set. A station not already in the file will be added only if ADDSTN is set. A time not already in the file will be added if there is room. If there is no room and CIRFLG is set, the earliest time in the file will be deleted.

RA_TMST (ISFFLN, DATTIM, STID, ADDSTN, CIRFLG, DATFLG, IRET)

Input parameters:

ISFFLN	INTEGER	Sounding file number
DATTIM	CHAR*15	Nominal date/time
STID	CHAR*	Station identifier
ADDSTN	LOGICAL	Add station flag
CIRFLG	LOGICAL	Circular file flag

Output parameters:

DATFLG	LOGICAL	Data already in file flag
IRET	INTEGER	Return code 0 = normal return -4 = time cannot be set -5 = station cannot be set

CHAPTER 22

UPPER-AIR DECODER (RU) LIBRARY

RU_ADJT	Adjust time to nearest 3 hours
RU_DCD2	Decode and write report
RU_DECD	Decode and write report
RU_GRPT	Get report from bulletin
RU_RTIM	Compute observation time
RU_SHDR	Decode upper-air header

UPPER-AIR DECODER (RU) LIBRARY

Upper-Air Decoder (RU) Library Summary

The RU library contains subroutines to decode and store upper-air reports. TT and PP reports of all types may be decoded. Output can be written to a GEMPAK unmerged sounding data set.

An upper-air decoder program must have access to bulletins of upper-air data. Given an upper-air bulletin, RU_GRPT extracts reports. RU_SHDR reads the report header, returning the station number and report day and hour. RU_DECD decodes the report and writes the output to a previously opened file. RU_DCD2 is a newer version of RU_DECD which includes the parameter ADDSTN. If ADDSTN is false, reporting stations not included in the output file will be added to it. RU_DECD always adds these stations to the file if there is room.

Several subroutines are available to process the time, since it is necessary to store a full date/time field in the data set. RU_RTIM combines a system or data reception time with the report day and hour to generate an observation time. RU_ADJT can be used to adjust the time to the nearest 3-hour interval.

The decoded output is in the order expected by an unmerged sounding file and by SN_WPRT, which writes the data to a file. Data from PPAA and PPCC reports will be merged with mandatory data or stored as mandatory data without temperature, dewpoint or height. Significant wind data are stored with a flag indicating whether the levels are height or pressure. If both height and pressure data are received for a station, only the most recent are saved.

ERROR MESSAGES:

In general, the errors encountered in the RU library are not fatal to continued execution, but just flag a problem with a particular report. Thus, it seems unlikely that a programmer will want to print a message every time the return code is non-zero. However, the return code values are summarized here as an aid to program development.

[RA -1]	No more input (substring, report, bulletin).
[RA -2]	No data found.
[RA -3]	Invalid character group.
[RA -4]	Error in setting station.
[RA -5]	End of report.
[RA xx-500]	Error xx in SN_GTIM.
[RA xx-600]	Error xx in SN_DTIM.
[RA xx-700]	Error xx in SN_ATIM after a time has been deleted.

23.34 SF_WSDD - WRITE TO SHIP FILE

This subroutine adds a station header and station data to a ship surface data file.

```
SF_WSDD ( ISFFLN, DATTIM, STID,   ISTNM, SLAT, SLON, SELV, STAT,
          COUN,   IHHMM,  SFDATA, IRET )
```

Input parameters:

ISFFLN	INTEGER	Surface file number
DATTIM	CHAR*	GEMPAK time
STID	CHAR*4	Station identifier
ISTNM	INTEGER	Station number
SLAT	REAL	Station latitude
SLON	REAL	Station longitude
SELV	REAL	Station elevation
STAT	CHAR*2	State
COUN	CHAR*2	Country
IHHMM	INTEGER	Station time (HHMM)
SFDATA (NPARM)	REAL	Station data

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -5 = too many reports -12 = DM error -20 = invalid time -21 = not single dim file

[RA xx-800] Error xx in SN_ATIM.
[RA xx-900] Error xx in SN_WPRT.

UPPER-AIR DECODER (RU) LIBRARY

RU Library Calls

RU_ADJT	(iotarr, / dattim, iret)
RU_DCD2	(isnfln, dattim, istnm, part, itopwn, wnknot, ihhmm, report, lenr, addstn, / irpnt, / iret)
RU_DECD	(isnfln, dattim, istnm, part, itopwn, wnknot, ihhmm, report, lenr, / irpnt, / iret)
RU_GRPT	(bultin, lenb, / ibpnt, / report, lenr, iret)
RU_RTIM	(isdtar, irday, irhour, / iodtar, iret)
RU_SHDR	(report, lenr, / irpnt, / part, istnm, iday, ihour, wnknot, itopwn, iret)

22.1 RU_ADJT - ADJUST TIME TO NEAREST 3 HOURS

This subroutine adjusts the time to the nearest 3-hourly interval.

RU_ADJT (IOTARR, DATTIM, IRET)

Input parameters:

IOTARR (5)	INTEGER	Observation time

Output parameters:

DATTIM	CHAR*	GEMPAK time rounded to 3 hours
IRET	INTEGER	Return code 0 = normal return

22.2 RU_DCD2 - DECODE AND WRITE REPORT

This subroutine decodes a single upper air report and writes the data to a file. The header information and DATTIM must be found before this subroutine is called. IRPNT must point to the first field after the header. This subroutine may be used for either real-time or archived data. Unlike RU_DECD, this subroutine has an input parameter ADDSTN which is a flag to indicate whether new reporting stations should be added to the file.

```
RU_DCD2 ( ISNFLN, DATTIM, ISTNM, PART, ITOPWN, WNKNOT, IHHMM,
          REPORT, LENR, ADDSTN, IRPNT, IRET )
```

Input parameters:

ISNFLN	INTEGER	Sounding file number
DATTIM	CHAR*15	Observation time
ISTNM	INTEGER	Station number
PART	CHAR*4	Part name
ITOPWN	INTEGER	Top level reporting winds
WNKNOT	LOGICAL	Flag for wind in knots
IHHMM	INTEGER	Actual hour/minute of report
REPORT	CHAR*	Report
LENR	INTEGER	Length of report
ADDSTN	LOGICAL	Add new stations flag

Input and output parameters:

IRPNT	INTEGER	Pointer to next field

Output parameters:

IRET	INTEGER	Return code 0 = normal return -2 = no data found -4 = error in setting station xx - 900 = error xx in SN_WPRT xx - 500 to xx - 800 see RU_TMST

22.3 RU_DECD - DECODE AND WRITE REPORT

This subroutine decodes a single upper air report and writes the data to a file. The header information and DATTIM must be found before this subroutine is called. IRPNT must point to the first field after the header. This subroutine may be used for either real-time or archived data. Note that this subroutine, unlike RU_DECD, has no ADDSTN option. Any reporting station which is not already in the sounding file will be added to it if there is room.

RU_DECD (ISNFLN, DATTIM, ISTNM, PART, ITOPWN, WNKNOT, IHHMM,
REPORT, LENR, IRPNT, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
DATTIM	CHAR*	Observation time
ISTNM	INTEGER	Station number
PART	CHAR*4	Part name
ITOPWN	INTEGER	Top level reporting winds
WNKNOT	LOGICAL	Flag for wind in knots
IHHMM	INTEGER	Actual hour/minute of report
REPORT	CHAR*	Report
LENR	INTEGER	Length of report

Input and output parameters:

IRPNT	INTEGER	Pointer to next field

Output parameters:

IRET	INTEGER	Return code 0 = normal return -2 = no data found -4 = error in setting station xx - 900 = error xx in SN_WPRT xx - 500 to xx - 800 see RU_TMST

22.4 RU_GRPT - GET REPORT FROM BULLETIN

This subroutine finds the next report in an upper-air bulletin. Upon entry, IBPNT points to the character to begin the search. The report returned will begin with TT, PP, or UU and will terminate with =, the start of another report, or the end of the bulletin.

RU_GRPT (BULTIN, LENB, IBPNT, REPORT, LENR, IRET)

Input parameters:

BULTIN	CHAR*	Upper-air bulletin
LENB	INTEGER	Length of bulletin

Input and output parameters:

IBPNT	INTEGER	Pointer in bulletin

Output parameters:

REPORT	CHAR*	Report
LENR	INTEGER	Length of report
IRET	INTEGER	Return code 0 = normal return -1 = no more reports

22.5 RU_RTIM - COMPUTE OBSERVATION TIME

This subroutine combines an integer system time and the report day and hour into an observation time. It is assumed that the system time accurately reflects the time at which the report was received, and is later than the actual report time (i.e., the time the observation was made).

RU_RTIM (ISDTAR, IRDAY, IRHOUR, IODTAR, IRET)

Input parameters:

ISDTAR (5)	INTEGER	System time
IRDAY	INTEGER	Report day
IRHOUR	INTEGER	Report hour

Output parameters:

IODTAR (5)	INTEGER	Observation date/time
IRET	INTEGER	Return code 0 = normal return

22.6 RU_SHDR - DECODE UPPER-AIR HEADER

This subroutine decodes the header from an upper-air TT or PP report. The fields after IRPNT are the part type, the station time and the station number. TOPWND is the hundreds digit for TTAA data and the tens digit for TTCC data. On return, IRPNT points to the first field after the station number.

RU_SHDR (REPORT, LENR, IRPNT, PART, ISTNM, IDAY, IHOUR, WNKNOT, ITOPWN, IRET)

Input parameters:

REPORT	CHAR*	Station report
LENR	INTEGER	Length of report

Input and output parameters:

IRPNT	INTEGER	Pointer within report

Output parameters:

PART	CHAR*4	Part name
ISTNM	INTEGER	Station number
IDAY	INTEGER	Observation day
IHOUR	INTEGER	Observation hour
WNKNOT	LOGICAL	Flag for speed in knots
ITOPWN	REAL	Pressure for last wind report
IRET	INTEGER	Return code 0 = normal return -2 = no report -3 = invalid group

CHAPTER 23

SURFACE (SF) LIBRARY

SF_ASTN	Add stations
SF_ATIM	Add time
SF_BEGS	Reset search
SF_CCLF	Create climate file
SF_CCLP	Create packed climate file
SF_CLOS	Close surface file
SF_CREF	Create standard file
SF_CRFP	Create packed standard file
SF_CSDF	Create ship file
SF_CSDP	Create packed ship file
SF_DDAT	Delete data
SF_DSTN	Delete station
SF_DTIM	Delete time
SF_FSTN	Find station
SF_FTIM	Find time
SF_GTIM	Get list of times
SF_OPNF	Open surface file
SF_OPNR	Open real-time surface file
SF_QDAT	Check for data
SF_QSTN	Get station information
SF_RDAT	Read data
SF_SNXT	Set next station
SF_SSTN	Set particular station
SF_STAT	Set a state/country
SF_STIM	Set time
SF_STNF	Add stations from table file
SF_STST	Get stations in state
SF_TNXT	Set next time
SF_TSTN	Set station
SF_TTIM	Set time
SF_UARE	Set station search
SF_USTN	Update station information
SF_WDAT	Write data
SF_WSDD	Write to ship file

Surface (SF) Library Summary

The surface library subroutines allow the programmer to access GEMPAK surface data files. Surface files contain meteorological observations from many locations for different times. The library contains modules which create and open files and read or write data to these files.

There are three types of surface files: standard, climate, and ship files. Standard files have stations as columns in the file and times in the rows. Climate data sets have stations in rows and times in the columns. Ship files, which are used for reports which are not from fixed locations, will have a single row with the station and time information in the same header. The type of file is determined by the subroutine used to create the file. Note that there are two subroutines for each type of file; one sends all the information about the file and the other reads a packing file to retrieve information about the parameters to be included in the file. Each of the three types of files can be opened and the data can be read using subroutines SF_OPNF and SF_RDAT. SF_WDAT can be used to write to standard or climate files; SF_WSDD is used to write to a ship file.

The following table shows the subroutines used with the three file types:

TYPE	CREATE	CREATE-PACK	WRITE
standard	SF_CREF	SF_CRFP	SF_WDAT
climate	SF_CCLF	SF_CCLP	SF_WDAT
ship	SF_CSDF	SF_CSDP	SF_WSDD

The subroutines to create or open a surface file return a file number which must be used in later subroutines to access the file.

The file GEMINC:GEMPRM.PRM contains the maximum values for array dimensions when using GEMPAK subroutines. A copy of this file has been included in the appendix for easy reference. MMFILE is the maximum number of files that can be open. LLMXTM is the maximum number of times that can be saved in a GEMPAK file. The maximum number of stations is LLSTFL and the maximum number of parameters is MMPARM.

After a file is opened, both the time and station must be selected before data can be read or written. There are two groups of subroutines that perform this function.

If data from many stations are to be accessed for a particular time, the time can be set using SF_STIM. The stations to be selected may be defined using LC_SARE or LC_UARE, which select stations using the GEMPAK variable, AREA. In addition, a new subroutine, SF_UARE, can be used to set a station search. This subroutine will allow programs to execute faster if a single station is to be found at a list of times. The search subroutines may be called before or after SF_STIM. Stations within the area are returned using SF_SNXT. The subroutines SF_SSTN, SF_STAT and SF_STST are included for compatibility with earlier versions of GEMPAK. Note that calls to these subroutines will delete searches already defined.

If data for many times at a particular station are required, the station may be selected using SF_TSTN. The time may then be defined using SF_TTIM. Alternatively, times may be returned using SF_TNXT.

All GEMPAK surface files contain information about the station in station headers. The station header names, contents, and the data type returned from the SF library are:

STID	Station identifier	CHARACTER*4
STNM	Station number	INTEGER
SLAT	Station latitude	REAL
SLON	Station longitude	REAL
SELV	Station elevation in meters	REAL
STAT	State	CHARACTER*2
COUN	Country	CHARACTER*2

Only SLAT and SLON are required for surface files. The other header variables are optional.

The subroutines SF_FTIM and SF_FSTN can be used to find a time and station in a data set. They will execute faster than the subroutines above, but can only be used with files where the times are in rows and the stations are in columns (or vice versa). They were designed to be used in real-time, data-ingest applications and should not be used for normal applications which use general surface files.

The parameter packing file specifies the parameters and packing information for a surface file. Each line must contain the following information separated by blanks or tabs:

parameter name	CHAR*4
minimum data value	REAL
maximum data value	REAL
resolution	REAL

The resolution should be an integral power of 10; otherwise the

next smaller resolution will be used (e.g., res = 0.5 will become 0.1). If the data are not to be packed, the minimum and maximum data values and the resolution should not be included. Note that either all of the parameters must have packing values or none of them must have them.

Some examples of subroutine sequences for accessing the data follow.

A sequence of subroutines to retrieve surface data for many stations at one time is:

```
Open the surface file              (SF_OPNF)
Define time                        (SF_STIM)
Define the area search             (LC_SARE)
Loop:
   Get the next station            (SF_SNXT)
   Read the data                   (SF_RDAT)
End loop
Close the file                     (SF_CLOS)
```

A sequence of subroutines to retrieve surface data for many times at one station is:

```
Initialize GEMPAK                  (IN_BDTA)

Open the surface file              (SF_OPNF)
Get times in file                  (SF_GTIM)
Get times to use                   (TI_FIND)
Set the station                    (SF_TSTN)
Loop:
   Get the next time               (SF_TTIM)
   Read the data                   (SF_RDAT)
End loop
Close the file                     (SF_CLOS)
```

ERROR MESSAGES:

```
[SF  +1]  There is no data at the station.
[SF  -1]  File ... could not be created.
[SF  -2]  File ... could not be opened.
[SF  -3]  File is not open.
[SF  -4]  No more times can be added.
[SF  -5]  No more stations can be added.
[SF  -6]  File is not a surface data file.
[SF  -7]  Station or time has not been set.
[SF  -8]  There are no more stations in file.
[SF  -9]  There are no more times in file.
[SF -10]  Station ... is not in file.
[SF -11]  Time ... is not in file.
[SF -12]  Error from DM library.
[SF -13]  Information cannot be deleted.
```

[SF -14] There are too many times in file.
[SF -15] A state/country search is invalid.
[SF -16] The station table file cannot be opened.
[SF -17] Time has not been set.
[SF -18] Station has not been set.
[SF -19] Cannot write to ship file.
[SF -20] Time ... is invalid.
[SF -21] File is not a ship file.

SF Library Calls

```
SF_ASTN  ( isffln, nstn, stid, istnm, slat, slon, selv, stat,
           coun, / nadd, iret )

SF_ATIM  ( isffln, dattim, / iret )

SF_BEGS  ( isffln, / iret )

SF_CCLF  ( filnam, iflsrc, nparm, parms, maxstn, maxtim, pkflg,
           iscale, iofset, ibits, stmflg, / isffln, iret )

SF_CCLP  ( filnam, prmfil, iflsrc, maxstn, maxtim, stmflg, / isffln,
           nparm, parms, pkflg, iret )

SF_CLOS  ( isffln, / iret )

SF_CREF  ( filnam, iflsrc, nparm, parms, maxstn, maxtim, pkflg,
           iscale, iofset, ibits, stmflg, / isffln, iret )

SF_CRFP  ( filnam, prmfil, iflsrc, maxstn, maxtim, stmflg, / isffln,
           nparm, parms, pkflg, iret )

SF_CSDF  ( filnam, iflsrc, nparm, parms, maxrpt, pkflg, iscale,
           iofset, ibits, stmflg, / isffln, iret )

SF_CSDP  ( filnam, prmfil, iflsrc, maxrpt, stmflg, / isffln, nparm,
           parms, pkflg, iret )

SF_DDAT  ( isffln, / iret )

SF_DSTN  ( isffln, stn, / iret )

SF_DTIM  ( isffln, dattim, / iret )

SF_FSTN  ( isffln, stn, / iret )

SF_FTIM  ( isffln, dattim, / iret )

SF_GTIM  ( isffln, maxtim, / ntime, timlst, iret )

SF_OPNF  ( filnam, wrtflg, / isffln, iflsrc, nparm, parms, iret )

SF_OPNR  ( filnam, / isffln, iflsrc, nparm, parms, iret )

SF_QDAT  ( isffln, / datflg, iret )

SF_QSTN  ( isffln, / stid, istnm, slat, slon, selv, stat, coun,
           iret )

SF_RDAT  ( isffln, / data, ihhmm, iret )
```

SURFACE (SF) LIBRARY

SF_SNXT (isffln, / stid, istnm, slat, slon, selv, iret)

SF_SSTN (isffln, stn, / stid, istnm, slat, slon, selv, iret)

SF_STAT (isffln, stcn, / iret)

SF_STIM (isffln, dattim, / iret)

SF_STNF (isffln, tbfile, / iret)

SF_STST (isffln, maxstn, stcn, / nstn, stid, istnm, iret)

SF_TNXT (isffln, / dattim, iret)

SF_TSTN (isffln, stn, / iret)

SF_TTIM (isffln, dattim, / iret)

SF_UARE (isffln, area, newfil, / arecur, / stn, iret)

SF_USTN (isffln, stid, istnm, slat, slon, selv, stat, coun, keynam, / iret)

SF_WDAT (isffln, ihhmm, data, / iret)

SF_WSDD (isffln, dattim, stid, istnm, slat, slon, selv, stat, coun, ihhmm, sfdata, / iret)

23.1 SF_ASTN - ADD STATIONS

This subroutine adds a list of stations to a surface data file. This subroutine can only be used if the times and stations are not mixed in row or column headers. NADD returns the number of stations actually added. This number may be less than NSTN if the file is full.

```
SF_ASTN  ( ISFFLN, NSTN, STID, ISTNM, SLAT, SLON, SELV, STAT,
           COUN,   NADD, IRET )
```

Input parameters:

ISFFLN	INTEGER	Surface file number
NSTN	INTEGER	Number of stations
STID (NSTN)	CHAR*4	Station identifiers
ISTNM (NSTN)	INTEGER	Station numbers
SLAT (NSTN)	REAL	Station latitudes
SLON (NSTN)	REAL	Station longitudes
SELV (NSTN)	REAL	Station elevations
STAT (NSTN)	CHAR*2	States
COUN (NSTN)	CHAR*2	Countries

Output parameters:

NADD	INTEGER	Number of stations added
IRET	INTEGER	Return code 0 = normal return -3 = file not open -5 = too many stations -19 = non-standard file

23.2 SF_ATIM - ADD TIME

This subroutine adds a time to a surface data file. This subroutine can only be used if times and stations are not mixed in row or column headers. If data are to be added for this time, SF_STIM must be called first.

SF_ATIM (ISFFLN, DATTIM, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
DATTIM	CHAR*	Date/time

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -4 = too many times -19 = non-standard file -20 = time is invalid

23.3 SF_BEGS - RESET SEARCH

This subroutine resets the search pointers to the beginning of a surface file. It does not change the time set by SF_STIM or the station set by SF_TSTN.

SF_BEGS (ISFFLN, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open

23.4 SF_CCLF - CREATE CLIMATE FILE

This subroutine creates a new climate surface data file. The file will store stations as rows of a DM file and times as columns.

If the packing flag, PKFLG, is set, data will be packed using values in ISCALE, IOFSET and IBITS. Note that SF_CRFP reads the parameters and packing information from a GEMPAK packing file.

If the station time flag is set, a single word is allocated with each data report to store the report time (HHMM). This time should be sent to SF_WDAT.

If the file cannot be created, error messages will be written.

The data source values are parameters in GEMINC:GEMPRM.PRM . These are not currently used by any GEMPAK program. Current definitions include:

MFNONE	unknown
MFAIRW	airways surface observation
MFMETR	Metar report
MFSHIP	ship report
MFBUOY	buoy report
MFSYNP	synoptic report

```
SF_CCLF  ( FILNAM, IFLSRC, NPARM, PARMS, MAXSTN, MAXTIM, PKFLG,
           ISCALE, IOFSET, IBITS, STMFLG, ISFFLN, IRET )
```

Input parameters:

FILNAM	CHAR*	Surface file name
IFLSRC	INTEGER	Data source
NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*4	Parameter names
MAXSTN	INTEGER	Maximum number of stations
MAXTIM	INTEGER	Maximum number of times
PKFLG	LOGICAL	Packing flag
ISCALE (NPARM)	INTEGER	Scaling factor
IOFSET (NPARM)	INTEGER	Offset term
IBITS (NPARM)	INTEGER	Number of bits
STMFLG	LOGICAL	Station time flag

Output parameters:

ISFFLN	INTEGER	Surface file number
IRET	INTEGER	Return code 0 = normal return -1 = file not created

23.5 SF_CCLP - CREATE PACKED CLIMATE FILE

This subroutine creates a new climate surface data file. The file will have times stored as columns of the DM file and stations as rows.

The contents of the file named in PRMFIL will determine the parameters to be contained in the data set and the packing, if any, to be used. PKFLG is set on output if the data are to be packed. All data packing and unpacking will be done internally. This subroutine is identical to SF_CCLF except that the packing information is read from a file in this subroutine. The data packing file is described in the introduction to this chapter.

If the station time flag is set, a single word is allocated with each data report to store the report time (HHMM). This time should be sent to SF_WDAT.

If the file cannot be created, error messages will be written.

The data source values are parameters in GEMINC:GEMPRM.PRM . These are not currently used by any GEMPAK program. Current definitions include:

MFNONE	unknown
MFAIRW	airways surface observation
MFMETR	Metar report
MFSHIP	ship report
MFBUOY	buoy report
MFSYNP	synoptic report

```
SF_CCLP  ( FILNAM, PRMFIL, IFLSRC, MAXSTN, MAXTIM, STMFLG, ISFFLN,
           NPARM,  PARMS,  PKFLG,  IRET )
```

Input parameters:

FILNAM	CHAR*	Surface file name
PRMFIL	CHAR*	Parameter packing file name
IFLSRC	INTEGER	Data source
MAXSTN	INTEGER	Maximum number of stations
MAXTIM	INTEGER	Maximum number of times
STMFLG	LOGICAL	Station time flag

Output parameters:

ISFFLN	INTEGER	Surface file number
NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*4	Parameter names
PKFLG	LOGICAL	Parameter packing flag
IRET	INTEGER	Return code 0 = normal return -1 = file not created

23.6 SF_CLOS - CLOSE SURFACE FILE

This subroutine closes a surface data file. This subroutine must be called to flush buffered data if anything has been written to the file.

SF_CLOS (ISFFLN, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -12 = DM error

23.7 SF_CREF - CREATE STANDARD FILE

This subroutine creates a new standard surface data file. The file will store times as rows of a DM file and stations as columns.

If the packing flag, PKFLG, is set, data will be packed using values in ISCALE, IOFSET and IBITS. Note that SF_CRFP reads the parameters and packing information from a GEMPAK packing file.

If the station time flag is set, a single word is allocated with each data report to store the report time (HHMM). This time should be sent to SF_WDAT.

If the file cannot be created, error messages will be written.

The data source values are parameters in GEMINC:GEMPRM.PRM . These are not currently used by any GEMPAK program. Current definitions include:

MFNONE	unknown
MFAIRW	airways surface observation
MFMETR	Metar report
MFSHIP	ship report
MFBUOY	buoy report
MFSYNP	synoptic report

```
SF_CREF ( FILNAM, IFLSRC, NPARM, PARMS, MAXSTN, MAXTIM, PKFLG,
          ISCALE, IOFSET, IBITS, STMFLG, ISFFLN, IRET )
```

Input parameters:

FILNAM	CHAR*	Surface file name
IFLSRC	INTEGER	Data source
NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*4	Parameter names
MAXSTN	INTEGER	Maximum number of stations
MAXTIM	INTEGER	Maximum number of times
PKFLG	LOGICAL	Packing flag
ISCALE (NPARM)	INTEGER	Scaling factor
IOFSET (NPARM)	INTEGER	Offset term
IBITS (NPARM)	INTEGER	Number of bits
STMFLG	LOGICAL	Station time flag

Output parameters:

ISFFLN	INTEGER	Surface file number
IRET	INTEGER	Return code 0 = normal return -1 = file not created

23.8 SF_CRFP - CREATE PACKED STANDARD FILE

This subroutine creates a new standard surface data file. The file will have times stored as rows of the DM file and stations as columns.

The contents of the file named in PRMFIL will determine the parameters to be contained in the data set and the packing, if any, to be used. PKFLG is set on output if the data are to be packed. All data packing and unpacking will be done internally. This subroutine is identical to SF_CREF except that the packing information is read from a file in this subroutine. The data packing file is described in the introduction to this chapter.

If the station time flag is set, a single word is allocated with each data report to store the report time (HHMM). This time should be sent to SF_WDAT.

If the file cannot be created, error messages will be written.

The data source values are parameters in GEMINC:GEMPRM.PRM. These are not currently used by any GEMPAK program. Current definitions include:

MFNONE	unknown
MFAIRW	airways surface observation
MFMETR	Metar report
MFSHIP	ship report
MFBUOY	buoy report
MFSYNP	synoptic report

```
SF_CRFP ( FILNAM, PRMFIL, IFLSRC, MAXSTN, MAXTIM, STMFLG, ISFFLN,
          NPARM,  PARMS,  PKFLG,  IRET )
```

Input parameters:

FILNAM	CHAR*	Surface file name
PRMFIL	CHAR*	Parameter packing file name
IFLSRC	INTEGER	Data source
MAXSTN	INTEGER	Maximum number of stations
MAXTIM	INTEGER	Maximum number of times
STMFLG	LOGICAL	Station time flag

Output parameters:

ISFFLN	INTEGER	Surface file number
NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*4	Parameter names
PKFLG	LOGICAL	Parameter packing flag
IRET	INTEGER	Return code 0 = normal return -1 = file not created

23.9 SF_CSDF - CREATE SHIP FILE

This subroutine creates a new ship surface data file. The file will store times and stations together as columns in row 1. This type of file may be used to store data, such as ship reports, where the station locations vary in time.

If the packing flag, PKFLG, is set, data will be packed using values in ISCALE, IOFSET and IBITS. Note that SF_CSDP reads the parameters and packing information from a GEMPAK packing file.

If the station time flag is set, a single word is allocated with each data report to store the report time (HHMM). This time should be sent with the report to SF_WSDD.

The subroutine SF_WSDD will write data to this file. The data can be read using SF_RDAT; all GEMPAK programs will be able to read this file.

If the file cannot be created, error messages will be written.

The data source values are parameters in GEMINC:GEMPRM.PRM . These are not currently used by any GEMPAK program. Current definitions include:

MFNONE	unknown
MFAIRW	airways surface observation
MFMETR	Metar report
MFSHIP	ship report
MFBUOY	buoy report
MFSYNP	synoptic report

```
SF_CSDF  ( FILNAM, IFLSRC, NPARM, PARMS,  MAXRPT, PKFLG,
           ISCALE, IOFSET, IBITS, STMFLG, ISFFLN, IRET )
```

Input parameters:

FILNAM		CHAR*	Surface file name
IFLSRC		INTEGER	Data source
NPARM		INTEGER	Number of parameters
PARMS	(NPARM)	CHAR*4	Parameter names
MAXRPT		INTEGER	Maximum number of reports
PKFLG		LOGICAL	Packing flag
ISCALE	(NPARM)	INTEGER	Scaling factor
IOFSET	(NPARM)	INTEGER	Offset term
IBITS	(NPARM)	INTEGER	Number of bits
STMFLG		LOGICAL	Station time flag

Output parameters:

ISFFLN	INTEGER	Surface file number
IRET	INTEGER	Return code

0 = normal return
-1 = file not created

23.10 SF_CSDP - CREATE PACKED SHIP FILE

This subroutine creates a new ship surface data file. The file will store times and stations together as columns in row 1. This type of file may be used to store data if the station locations vary, such as for ship reports.

The parameter packing file named in PRMFIL will determine the parameters to be contained in the data set and the packing, if any, to be used. PKFLG is set on output if the data are to be packed. All data packing and unpacking will be done internally. This subroutine is identical to SF_CSDF except that the packing information is read from a packing file in this subroutine. The data packing file is described in the introduction to this chapter.

If the station time flag is set, a single word is allocated with each data report to store the report time (HHMM). This time should be sent with the report.

The subroutine SF_WSDD will write data to this file. The data can be read using SF_RDAT, so all GEMPAK programs will be able to read this file.

The data source values are parameters in GEMINC:GEMPRM.PRM . These are not currently used by any GEMPAK program. Current definitions include:

MFNONE	unknown
MFAIRW	airways surface observation
MFMETR	Metar report
MFSHIP	ship report
MFBUOY	buoy report
MFSYNP	synoptic report

SF_CSDP (FILNAM, PRMFIL, IFLSRC, MAXRPT, STMFLG, ISFFLN, NPARM, PARMS, PKFLG, IRET)

Input parameters:

FILNAM	CHAR*	Surface file name
PRMFIL	CHAR*	Parameter packing file name
IFLSRC	INTEGER	Data source
MAXRPT	INTEGER	Maximum number of reports
STMFLG	LOGICAL	Station time flag

Output parameters:

ISFFLN	INTEGER	Surface file number
NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*4	Parameter names
PKFLG	LOGICAL	Parameter packing flag
IRET	INTEGER	Return code

0 = normal return
-1 = file not created

23.11 SF_DDAT - DELETE DATA

This subroutine deletes data for a particular station and time from a surface data file. The time and station must be set before calling this subroutine.

SF_DDAT (ISFFLN, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -7 = location not set -12 = DM error

23.12 SF_DSTN - DELETE STATION

This subroutine deletes a station from a surface file. All the data corresponding to the station will be deleted along with the station header.

SF_DSTN (ISFFLN, STN, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
STN	CHAR*	Station number or id

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -13 = delete error

23.13 SF_DTIM - DELETE TIME

This subroutine deletes a time from a surface file. All the data corresponding to the time will be deleted along with the header storing the time.

SF_DTIM (ISFFLN, DATTIM, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
DATTIM	CHAR*	GEMPAK date/time

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -13 = delete error

23.14 SF_FSTN - FIND STATION

This subroutine finds the location of the specified station in a file. The first occurrence of the station is saved. This subroutine may only be used when times and stations are not mixed in row or column headers in the file. It will execute faster than the SF_Sxxx or SF_Txxx subroutines, but is intended to be used only for real-time ingest programs where the structure of the file is known by the programmer. The time may be set using SF_FTIM. These subroutines may be called in either order.

SF_FSTN (ISFFLN, STN, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
STN	CHAR*	Station number or id

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -10 = station not found -19 = non-standard file

23.15 SF_FTIM - FIND TIME

This subroutine finds the location of the specified date/time in a file. The first occurrence containing the time is saved. This subroutine may only be used when times and stations are not mixed in row or column headers in the file. It will execute faster than the SF_Sxxx or SF_Txxx subroutines, but is intended to be used only for real-time ingest programs where the structure of the file is known by the programmer. The station may be set using SF_FSTN. These subroutines may be called in either order.

SF_FTIM (ISFFLN, DATTIM, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
DATTIM	CHAR*	Date/time

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -11 = time not found -19 = non-standard file

23.16 SF_GTIM - GET LIST OF TIMES

This subroutine returns a list of times available in a surface data file. The times are ordered from the earliest to the latest.

SF_GTIM (ISFFLN, MAXTIM, NTIME, TIMLST, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
MAXTIM	INTEGER	Maximum number of times

Output parameters:

NTIME	INTEGER	Number of times returned
TIMLST (NTIME)	CHAR*	GEMPAK times
IRET	INTEGER	Return code 0 = normal return -3 = file not open -14 = too many times in file

23.17 SF_OPNF - OPEN SURFACE FILE

This subroutine opens an existing surface data file.

SF_OPNF (FILNAM, WRTFLG, ISFFLN, IFLSRC, NPARM, PARMS, IRET)

Input parameters:

FILNAM	CHAR*	Surface file name
WRTFLG	LOGICAL	Write access flag

Output parameters:

ISFFLN	INTEGER	File number
IFLSRC	INTEGER	Data source
NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*4	Parameter names
IRET	INTEGER	Return code 0 = normal return -2 = file could not be opened -6 = file not surface file -22 = file name is blank

C-6

23.18 SF_OPNR - OPEN REAL-TIME SURFACE FILE

This subroutine opens an existing surface data file for real-time data ingest. The file is opened for shared write access. This subroutine should not be used for non-real-time applications.

SF_OPNR (FILNAM, ISFFLN, IFLSRC, NPARM, PARMS, IRET)

Input parameters:

FILNAM	CHAR*	Surface file name

Output parameters:

ISFFLN	INTEGER	File number
IFLSRC	INTEGER	Data source
NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*4	Parameter names
IRET	INTEGER	Return code 0 = normal return -2 = file could not be opened -6 = file not surface file -22 = file name is blank

23.19 SF_QDAT - CHECK FOR DATA

This subroutine sets a flag indicating whether data for the current station and time are stored in a file.

SF_QDAT (ISFFLN, DATFLG, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number

Output parameters:

DATFLG	LOGICAL	Data present flag
IRET	INTEGER	Return code 0 = normal return -3 = file not open -7 = location not set

23.20 SF_QSTN - GET STATION INFORMATION

This subroutine gets station information for the current station. Both the time and station must be set before this subroutine is called.

SF_QSTN (ISFFLN, STID, ISTNM, SLAT, SLON, SELV, STAT, COUN,
IRET)

Input parameters:

ISFFLN	INTEGER	Sounding file number

Output parameters:

STID	CHAR*4	Station identifier
ISTNM	INTEGER	Station number
SLAT	REAL	Station latitude
SLON	REAL	Station longitude
SELV	REAL	Station elevation
STAT	CHAR*2	State
COUN	CHAR*2	Country
IRET	INTEGER	Return code 0 = normal return -4 = file not open -7 = location not set

23.21 SF_RDAT - READ DATA

This subroutine reads data from a surface data file. The time and station must be set before calling this subroutine.

SF_RDAT (ISFFLN, DATA, IHHMM, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number

Output parameters:

DATA (NPARM)	REAL	Station data
IHHMM	INTEGER	Station hour and minute
IRET	INTEGER	Return code 1 = no data at station 0 = normal return -3 = file not open -7 = location not set

23.22 SF_SNXT - SET NEXT STATION

This subroutine selects the next station in a surface file. SF_STIM must be called to set the time before this subroutine is called. Stations to be found can be set in SF_UARE. Data for this station may be read or written by calling SF_RDAT or SF_WDAT, respectively.

SF_SNXT (ISFFLN, STID, ISTNM, SLAT, SLON, SELV, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number

Output parameters:

STID	CHAR*4	Station identifier
ISTNM	INTEGER	Station number
SLAT	REAL	Station latitude
SLON	REAL	Station longitude
SELV	REAL	Station elevation
IRET	INTEGER	Return code 0 = normal return -3 = file not open -8 = no more stations -17 = time not set

23.23 SF_SSTN - SET PARTICULAR STATION

This subroutine selects a station in a surface file. SF_STIM must be called before this subroutine is called. This subroutine will delete any searches previously set. Data for this station can be read or written by calling SF_RDAT or SF_WDAT, respectively.

SF_SSTN (ISFFLN, STN, STID, ISTNM, SLAT, SLON, SELV, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
STN	CHAR*	Station id or number

Output parameters:

STID	CHAR*4	Station identifier
ISTNM	INTEGER	Station number
SLAT	REAL	Station latitude
SLON	REAL	Station longitude
SELV	REAL	Station elevation
IRET	INTEGER	Return code 0 = normal return -3 = file not open -10 = station not in file -17 = time not set

23.24 SF_STAT - SET A STATE/COUNTRY

This subroutine selects a state or country. Later calls to SF_SNXT will return stations in the state or country. SF_STIM must be called before this subroutine is called. This subroutine is included for compatibility with earlier versions of GEMPAK. A search for stations should be set using SF_UARE.

SF_STAT (ISFFLN, STCN, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
STCN	CHAR*2	State or country abbreviation

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -15 = no state/country info

23.25 SF_STIM - SET TIME

This subroutine sets the time in a surface file. All later station searches will return stations corresponding to this time.

SF_STIM (ISFFLN, DATTIM, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
DATTIM	CHAR*	GEMPAK date/time

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -11 = time not in file -17 = time not set

23.26 SF_STNF - ADD STATIONS FROM TABLE FILE

This subroutine adds stations from a table file to a surface file. This subroutine can only be used if the times and stations are not mixed in row and column headers.

SF_STNF (ISFFLN, TBFILE, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
TBFILE	CHAR*	Station table file name

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -5 = no more space -16 = station file not opened -19 = non-standard file

23.27 SF_STST - GET STATIONS IN STATE

This subroutine returns a list of stations in a state or country. SF_STIM must be called before this subroutine is called. This subroutine is included for compatibility with earlier versions of GEMPAK. A search for stations should be set using SF_UARE.

SF_STST (ISFFLN, MAXSTN, STCN, NSTN, STID, ISTNM, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
MAXSTN	INTEGER	Maximum number of stations
STCN	CHAR*2	State/country name

Output parameters:

NSTN	INTEGER	Number of stations
STID (NSTN)	CHAR*4	Station identifiers
ISTNM (NSTN)	INTEGER	Station numbers
IRET	INTEGER	Return code 2 = too many stations 0 = normal return -3 = file not open -15 = invalid search

23.28 SF_TNXT - SET NEXT TIME

This subroutine selects the next time in a surface file. SF_TSTN must be called to set the station before this subroutine is called. The times will be returned in the order in which they appear in the file, rather than in chronological order. Data for this time can be read or written by calling SF_RDAT or SF_WDAT, respectively.

SF_TNXT (ISFFLN, DATTIM, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number

Output parameters:

DATTIM	CHAR*	GEMPAK date/time
IRET	INTEGER	Return code
		0 = normal return
		-3 = file not open
		-9 = no more times
		-18 = station not set

23.29 SF_TSTN - SET STATION

This subroutine sets the station in a surface file. All later time searches will return times corresponding to this station.

SF_TSTN (ISFFLN, STN, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
STN	CHAR*	Station number or id

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -10 = station not in file

23.30 SF_TTIM - SET TIME

This subroutine sets the time in a surface file. SF_TSTN must be called before this subroutine is called. Data for this time can be read or written by calling SF_RDAT or SF_WDAT, respectively.

SF_TTIM (ISFFLN, DATTIM, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
DATTIM	CHAR*	GEMPAK date/time

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -11 = time not found -18 = station not set

23.31 SF_UARE - SET STATION SEARCH

This subroutine sets the search criteria in a surface file using the value for AREA input by the user. The area may be composed of subareas which are separated by slashes (/). This subroutine will be more efficient than the equivalent LC_UARE when searching for a single station at multiple times. If the search is not for a single station, the appropriate calls to the LC library will be made.

SF_UARE (ISFFLN, AREA, NEWFIL, ARECUR, STN, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
AREA	CHAR*	Area to be defined
NEWFIL	LOGICAL	New file flag

Input and output parameters:

ARECUR	CHAR*	Current area

Output parameters:

STN	CHAR*	Center station name
IRET	INTEGER	Return code 0 = normal return -1 = invalid area name

23.32 SF_USTN - UPDATE STATION INFORMATION

This subroutine updates the header information for a station in a surface data file. This subroutine can only be used if the times and stations are not mixed in row or column headers.

```
SF_USTN ( ISFFLN, STID, ISTNM, SLAT, SLON, SELV, STAT,
          COUN, KEYNAM, IRET )
```

Input parameters:

ISFFLN	INTEGER	Surface file number
STID	CHAR*4	Station identifier
ISTNM	INTEGER	Station number
SLAT	REAL	Station latitude
SLON	REAL	Station longitude
SELV	REAL	Station elevation
STAT	CHAR*2	State
COUN	CHAR*2	Country
KEYNAM	CHAR*4	Key to update (STID or STNM)

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -10 = station not in file -12 = DM error -19 = non-standard file

23.33 SF_WDAT - WRITE DATA

This subroutine writes data to a surface data file. The time and station must both be set before this subroutine is called. The station time will be stored if the station time flag, STMFLG, was set when the file was created.

SF_WDAT (ISFFLN, IHHMM, DATA, IRET)

Input parameters:

ISFFLN	INTEGER	Surface file number
IHHMM	INTEGER	Station time (HHMM)
DATA (NPARM)	REAL	Surface data array

Output parameters:

IRET	INTEGER	Return code 0 = normal return -3 = file not open -7 = location not set -12 = DM error

CHAPTER 24

SOUNDING (SN) LIBRARY

SN_ASTN	Add station
SN_ATIM	Add time
SN_BEGS	Reset search
SN_CLOS	Close sounding file
SN_CREF	Create sounding file
SN_CRFP	Create using packing file
SN_CRUA	Create unmerged file
SN_DDAT	Delete data
SN_DSTN	Delete station
SN_DTIM	Delete time
SN_FSTN	Find station
SN_FTIM	Find time
SN_GTIM	Get list of times
SN_MAND	Set mandatory flag
SN_MTYP	Define data merge type
SN_OPNF	Open sounding file
SN_OPNR	Open real-time sounding file
SN_QDAT	Check for data
SN_QSTN	Get station information
SN_RDAT	Read data
SN_RPRT	Read part from unmerged file
SN_RTYP	Get level types
SN_SNXT	Get next station
SN_SSTN	Set particular station
SN_STIM	Set time for station search
SN_STNF	Add stations from table file
SN_TNXT	Get next station
SN_TSTN	Set station search
SN_TTIM	Set particular time
SN_USTN	Update station information
SN_WDAT	Write data
SN_WPRT	Write part to unmerged file

Sounding (SN) Library Summary

The sounding library subroutines allow the programmer to access GEMPAK upper-air data files. These files contain meteorological observations from many locations for different times. The library contains modules which create and open files and read or write data to these files.

There are two types of GEMPAK sounding files: merged and unmerged. Merged files may contain an arbitrary set of parameters which report at every level. Unmerged files store mandatory and significant data separately in the following parts with the given parameters:

TTAA	mandatory data below 100 mb	PRES TEMP DWPT DRCT SPED HGHT
TTBB	sig temp data below 100 mb	PRES TEMP DWPT
PPBB	sig wind data below 100 mb	HGHT DRCT SPED or PRES DRCT SPED
TTCC	mandatory data above 100 mb	PRES TEMP DWPT DRCT SPED HGHT
TTDD	sig temp data above 100 mb	PRES TEMP DWPT
PPDD	sig wind data above 100 mb	HGHT DRCT SPED or PRES DRCT SPED

When wind data appear on pressure surfaces, the first pressure is set to the negative of its value as a flag.

Data that are to be written to an unmerged file must be in the specified order. When data are returned from an unmerged file, data from all the parts will be merged. Interpolation will be used to fill in the significant data levels.

Merged data files can be created using SN_CREF or SN_CRFP; unmerged files can be created using SN_CRUA. SN_OPNF will open either file type. SN_RDAT will read data from all files; unmerged data will be returned as a merged data set. SN_RTYP can be called to determine whether each level is mandatory, significant temperature, or significant wind level data. SN_MAND can be called to request that only mandatory data below 100 mb be returned when SN_RDAT is called. SN_WDAT writes to merged files; SN_WPRT writes to unmerged files.

The subroutines to create or open a sounding file return a file number which must be used in later subroutines to access the file.

The file GEMINC:GEMPRM.PRM contains the maximum values for array dimensions when using GEMPAK subroutines. A copy of this file has been included in the appendix for easy reference. MMFILE is the maximum number of files that can be open. LLMXTM is the maximum number of times that can be saved in a GEMPAK5 file. The maximum

number of stations is LLSTFL and the maximum number of parameters is MMPARM.

After a file is opened, both the time and station must be selected before data can be read or written. There are two groups of subroutines that perform this function.

If data from many stations are to be accessed for a particular time, the time can be set using SN_STIM. The stations to be selected may be defined using LC_SARE or LC_UARE, which select stations using the GEMPAK variable, AREA. The LC subroutines may be called before or after SN_STIM. Stations within the area are returned using SN_SNXT.

If data for many times at a particular station are required, the station may be selected using SN_TSTN. The time may then be defined using SN_TTIM. Alternatively, times may be returned using SN_TNXT.

All GEMPAK files contain information about the station in station headers. The station header names, contents, and the data types returned from the SN library are:

STID	Station identifier	CHARACTER*4
STNM	Station number	INTEGER
SLAT	Station latitude	REAL
SLON	Station longitude	REAL
SELV	Station elevation in meters	REAL
STAT	State	CHARACTER*2
COUN	Country	CHARACTER*2

Only SLAT and SLON are required for sounding files. The other header variables are optional.

The subroutines SN_FTIM and SN_FSTN can be used to find a time and station in a data set. They will execute faster than the subroutines above, but can only be used with files where the times are in rows and the stations are in columns (or vice versa). They were designed to be used in real-time data ingest applications and should not be used for normal applications which use general sounding files.

Some examples of subroutine sequences for accessing the data follow.

A sequence of subroutines to retrieve sounding data for many stations at one time is:

Initialize GEMPAK	(IN_BDTA)
Open the file	(SN_OPNF)
Define the time	(SN_STIM)

```
    Define the area search                      (LC_SARE)
    Loop:
       Get the next station                     (SN_SNXT)
       Read the data                            (SN_RDAT)
    End loop
    Close the file                              (SN_CLOS)
```

A sequence of subroutines to retrieve sounding data for many times at one station is:

```
    Open the sounding file                      (SN_OPNF)
    Get times in file                           (SN_GTIM)
    Get times to use                            (TI_FIND)
    Set the station                             (SN_TSTN)
    Loop:
       Get the next time                        (SN_TTIM)
       Read the data                            (SN_RDAT)
    End loop
    Close the file                              (SN_CLOS)
```

ERROR MESSAGES:

```
[SN   1]  No data at station.
[SN  -1]  File ... could not be created.
[SN  -2]  File ... could not be opened.
[SN  -3]  Invalid part type.
[SN  -4]  File not open.
[SN  -5]  No more times can be added.
[SN  -6]  No more stations can be added.
[SN  -7]  File ... is not a sounding data set.
[SN  -8]  Station or time has not been set.
[SN  -9]  No more stations in file.
[SN -10]  No more times in file.
[SN -11]  Station ... is not in file.
[SN -12]  Time ... is not in file.
[SN -13]  Data management (DM) error.
[SN -14]  Too many times in file.
[SN -15]  Delete error.
[SN -16]  Too many levels at station.
[SN -17]  Invalid merged/unmerged data file.
[SN -18]  Station table file cannot be opened.
[SN -19]  Time has not been set.
[SN -20]  Station has not been set.
[SN -21]  Non-standard sounding station.
[SN -22]  Invalid part name.
[SN -23]  Time ... is invalid.
[SN -24]  File name is blank.
```

SN Library Calls

SN_ASTN (isnfln, nstn, stid, istnm, slat, slon, selv, stat, coun, / nadd, iret)

SN_ATIM (isnfln, dattim, / iret)

SN_BEGS (isnfln, / iret)

SN_CLOS (isnfln, / iret)

SN_CREF (filnam, iflsrc, nparm, parms, maxstn, maxtim, pkflg, iscale, iofset, ibits, stmflg, / isnfln, iret)

SN_CRFP (filnam, prmfil, iflsrc, maxstn, maxtim, stmflg, / isnfln, nparm, parms, pkflg, iret)

SN_CRUA (filnam, iflsrc, iptype, maxstn, maxtim, pkflg, stmflg, trpflg, / isnfln, iret)

SN_DDAT (isnfln, / iret)

SN_DSTN (isnfln, stn, / iret)

SN_DTIM (isnfln, dattim, / iret)

SN_FSTN (isnfln, stid, / iret)

SN_FTIM (isnfln, dattim, / iret)

SN_GTIM (isnfln, maxtim, / ntime, timlst, iret)

SN_MAND (isnfln, mandat, / iret)

SN_MTYP (isnfln, iztype, / iret)

SN_OPNF (filnam, wrtflg, / isnfln, iflsrc, nparm, parms, ivert, mrgdat, iret)

SN_OPNR (filnam, / isnfln, iflsrc, nparm, parms, ivert, mrgdat, iret)

SN_QDAT (isnfln, / datflg, iret)

SN_QSTN (isnfln, / stid, istnm, slat, slon, selv, stat, coun, iret)

SN_RDAT (isnfln, / nlev, data, ihhmm, iret)

SN_RPRT (isnfln, part, / ihhmm, nlev, data, zwind, iret)

SN_RTYP (isnfln, / nlev, idtype, iret)

SN_SNXT (isnfln, / stid, istnm, slat, slon, selv, iret)

SN_SSTN (isnfln, stn, / stid, istnm, slat, slon, selv, iret)

SN_STIM (isnfln, dattim, / iret)

SN_STNF (isnfln, tbfile, / iret)

SN_TNXT (isnfln, / dattim, iret)

SN_TSTN (isnfln, stn, / iret)

SN_TTIM (isnfln, dattim, / iret)

SN_USTN (isnfln, stid, istnm, slat, slon, selv, stat, coun, keynam, / iret)

SN_WDAT (isnfln, ihhmm, nlev, data, / iret)

SN_WPRT (isnfln, part, ihhmm, nlev, data, zwind, / iret)

24.1 SN_ASTN - ADD STATION

This subroutine adds a list of stations to a sounding data file. This subroutine can only be used if the times and stations are not mixed in row or column headers.

```
SN_ASTN ( ISNFLN, NSTN, STID, ISTNM, SLAT, SLON, SELV, STAT,
          COUN, NADD, IRET )
```

Input parameters:

ISNFLN	INTEGER	Sounding file number
NSTN	INTEGER	Number of stations
STID (NSTN)	CHAR*4	Station identifiers
ISTNM (NSTN)	INTEGER	Station numbers
SLAT (NSTN)	REAL	Station latitudes
SLON (NSTN)	REAL	Station longitudes
SELV (NSTN)	REAL	Station elevations
STAT (NSTN)	CHAR*2	States
COUN (NSTN)	CHAR*2	Countries

Output parameters:

NADD	INTEGER	Number of stations added
IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = too many stations -21 = non-standard file

24.2 SN_ATIM - ADD TIME

This subroutine adds a time to a sounding data file. This subroutine can only be used if times and stations are not mixed in row or column headers.

SN_ATIM (ISNFLN, DATTIM, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
DATTIM	CHAR*	Date/time

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -5 = time cannot be added -21 = non-standard file -23 = time is invalid

24.3 SN_BEGS - RESET SEARCH

This subroutine resets the search pointers to the beginning of a sounding file. It does not reset the time set by SN_STIM or the station set by SN_TSTN.

SN_BEGS (ISNFLN, IRET)

Input parameters:		
ISNFLN	INTEGER	Sounding file number

Output parameters:		
IRET	INTEGER	Return code 0 = normal return -4 = file not open

24.4 SN_CLOS - CLOSE SOUNDING FILE

This subroutine closes a sounding data file. This subroutine must be called to flush local data buffers if anything has been written to the file.

SN_CLOS (ISNFLN, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -13 = DM error

24.5 SN_CREF - CREATE SOUNDING FILE

This subroutine creates a new standard sounding data file. The file will store times as rows of a DM file and stations as columns. If the packing flag, PKFLG, is set, data will be packed using values in ISCALE, IOFSET and IBITS. Note that SN_CRFP gets packing information from a file. If the station time flag is set, a single word is allocated with each data report to store the report time (HHMM). This time should be sent to SN_WDAT.

The data source values are parameters in GEMINC:GEMPRM.PRM.

```
SN_CREF  ( FILNAM, IFLSRC, NPARM, PARMS, MAXSTN, MAXTIM, PKFLG,
           ISCALE, IOFSET, IBITS, STMFLG, ISNFLN, IRET)
```

Input parameters:

FILNAM		CHAR*	Surface file name
IFLSRC		INTEGER	Data source
NPARM		INTEGER	Number of parameters
PARMS	(NPARM)	CHAR*4	Parameter names
MAXSTN		INTEGER	Maximum number of stations
MAXTIM		INTEGER	Maximum number of times
PKFLG		LOGICAL	Packing flag
ISCALE	(NPARM)	INTEGER	Scaling factor
IOFSET	(NPARM)	INTEGER	Offset term
IBITS	(NPARM)	INTEGER	Number of bits
STMFLG		LOGICAL	Station time flag

Output parameters:

ISNFLN	INTEGER	Sounding file number
IRET	INTEGER	Return code 0 = normal return -1 = file not created

24.6 SN_CRFP - CREATE USING PACKING FILE

This subroutine creates a new sounding data file. The file will have times stored as rows of the DM file and stations stored as columns. If the station time flag is set, a single word is allocated with each data report to store the report time (HHMM). This time should be sent to SN_WDAT. The parameter packing flag is set if the data will be packed internally according to the format in PRMFIL. Note that SN_CREF includes the packing information in its arguments.

The data source values are parameters in GEMINC:GEMPRM.PRM.

```
SN_CREF  ( FILNAM, PRMFIL, IFLSRC, MAXSTN, MAXTIM, STMFLG, ISNFLN,
           NPARM,  PARMS,  PKFLG,  IRET )
```

Input parameters:

FILNAM	CHAR*	Sounding file name
PRMFIL	CHAR*	Parameter packing file name
IFLSRC	INTEGER	Data source
MAXSTN	INTEGER	Maximum number of stations
MAXTIM	INTEGER	Maximum number of times
STMFLG	LOGICAL	Station time flag

Output parameters:

ISNFLN	INTEGER	Sounding file number
NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*4	Parameter names
PKFLG	LOGICAL	Parameter packing flag
IRET	INTEGER	Return code 0 = normal return -1 = file not created

Parameter packing file:

This file specifies the parameters and packing information for a sounding file. Each line must contain the following information separated by blanks or tabs:

parameter name	CHAR*4
minimum data value	REAL
maximum data value	REAL
resolution	REAL

The resolution should be an integral value of 10; otherwise, the next smaller resolution will be used (e.g. res = .5 will become .1). If the data are not to be packed, the minimum and maximum data values and the resolution should not be included. Note that either all of the parameters or none of the parameters must have packing information.

24.7 SN_CRUA - CREATE UNMERGED FILE

This subroutine creates a sounding data file which has mandatory and significant data stored separately. If the packing flag, PKFLG, is set, data will be packed using standard packing values. If the station time flag is set, a single word is allocated with each data report to store the report time (HHMM). This time should be sent to SN_WPRT. TRPFLG is used to store tropopause data and is not implemented yet.

The data source values are parameters in GEMINC:GEMPRM.PRM.

```
SN_CRUA  ( FILNAM, IFLSRC, IPTYPE, MAXSTN, MAXTIM, PKFLG, STMFLG,
           TRPFLG, ISNFLN, IRET )
```

Input parameters:

FILNAM	CHAR*	Sounding file name
IFLSRC	INTEGER	Data source
IPTYPE	INTEGER	Data parts to be stored: 1 = man below 100 mb 2 = man & sig below 100 mb 3 = man & sig below & above
MAXSTN	INTEGER	Maximum number of stations
MAXTIM	INTEGER	Maximum number of times
PKFLG	LOGICAL	Packing flag
STMFLG	LOGICAL	Station time flag
TRPFLG	LOGICAL	Tropopause flag

Output parameters:

ISNFLN	INTEGER	Sounding file number
IRET	INTEGER	Return code 0 = normal return -1 = file not created -22 = invalid part type

24.8 SN_DDAT - DELETE DATA

This subroutine deletes data for a particular station and time from a sounding data file. The time and station must be set before calling this subroutine.

SN_DDAT (ISNFLN, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number

Output parameters:

IRET	INTEGER	Return code 0 = normal return -7 = location not set -15 = delete error

24.9 SN_DSTN - DELETE STATION

This subroutine deletes a station from a sounding file. All the data corresponding to the station will be deleted along with the station headers.

SN_DSTN (ISNFLN, STN, IRET)

Input parameters:		
ISNFLN	INTEGER	Sounding file number
STN	CHAR*	Station number or id
Output parameters:		
IRET	INTEGER	Return code 0 = normal return -4 = file not open -15 = delete error

24.10 SN_DTIM - DELETE TIME

This subroutine deletes a time from a sounding file. All the data corresponding to the time will be deleted along with the headers storing the time.

SN_DTIM (ISNFLN, DATTIM, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
DATTIM	CHAR*	GEMPAK date/time

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -15 = delete error

24.11 SN_FSTN - FIND STATION

This subroutine finds the location of the specified station in a DM file. The first row or column containing the station is set in the common area. This subroutine may only be used when times and stations are not mixed in row or column headers in the file. It will execute faster than the SN_Sxxx or SN_Txxx subroutines, but is intended to be used only for real-time ingest programs where the structure of the file is known by the programmer. The time may be set using SN_FTIM. These subroutines may be called in either order.

SN_FSTN (ISNFLN, STID, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
STID	CHAR*	Station number or id

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -11 = station not found -21 = non-standard file

24.12 SN_FTIM - FIND TIME

This subroutine finds the location of the specified date/time in a DM file. The first row or column containing the time is set in the common area. This subroutine may only be used when times and stations are not mixed in row or column headers in the file. It will execute faster than the SN_Sxxx or SN_Txxx subroutines, but is intended to be used only for real-time ingest programs where the structure of the file is known by the programmer. The station may be set using SN_FSTN. These subroutines may be called in either order.

SN_FTIM (ISNFLN, DATTIM, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
DATTIM	CHAR*	Date/time

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -15 = time not set -21 = non-standard file

24.13 SN_GTIM - GET LIST OF TIMES

This subroutine returns a list of times available in a sounding data file. The times are ordered from the earliest to the latest.

SN_GTIM (ISNFLN, MAXTIM, NTIME, TIMLST, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
MAXTIM	INTEGER	Maximum number of times

Output parameters:

NTIME	INTEGER	Number of times returned
TIMLST (NTIME)	CHAR*	GEMPAK times
IRET	INTEGER	Return code 0 = normal return -4 = file not open -14 = too many times in file

24.14 SN_MAND - SET MANDATORY FLAG

This subroutine allows the user to set a flag requesting that only mandatory data be returned when the upper-air dataset is an unmerged file. The default is to merge all data.

SN_MAND (ISNFLN, MANDAT, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
MANDAT	LOGICAL	Only mandatory data flag

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open

24.15 SN_MTYP - DEFINE DATA MERGE TYPE

This subroutine specifies the type of interpolation to be used for the height field in an unmerged data set. This interpolation adds heights to significant temperature levels. The default merge type is 3. If IZTYPE is 1, the height will be interpolated with respect to the logarithm of pressure. If IZTYPE is 2, the moist hydrostatic height field is computed. If IZTYPE is 3, the moist hydrostatic field is computed as in 2, but is scaled to retain the height values received with the mandatory data.

```
SN_MTYP ( ISNFLN, IZTYPE, IRET )
```

Input parameters:

ISNFLN	INTEGER	Sounding file number
IZTYPE	INTEGER	Type of height interpolation 1 = int wrt log p 2 = moist hydrostatic comp 3 = scaled moist hydro comp

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open

24.16 SN_OPNF - OPEN SOUNDING FILE

This subroutine opens an existing sounding data file.

SN_OPNF (FILNAM, WRTFLG, ISNFLN, IFLSRC, NPARM, PARMS, IVERT, MRGDAT, IRET)

Input parameters:

FILNAM	CHAR*	Sounding file name
WRTFLG	LOGICAL	Write access flag

Output parameters:

ISNFLN	INTEGER	File number
IFLSRC	INTEGER	Data source
NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*4	Parameter names
IVERT	INTEGER	Vertical coordinate
MRGDAT	LOGICAL	Merged data flag
IRET	INTEGER	Return code 0 = normal return -2 = file could not be opened -7 = file not sounding file -24 = file name is blank

24.17 SN_OPNR - OPEN REAL-TIME SOUNDING FILE

This subroutine opens an existing sounding data file for real-time data ingest. The file is opened for shared write access. This subroutine should not be used for non-real-time applications.

SN_OPNR (FILNAM, ISNFLN, IFLSRC, NPARM, PARMS, IVERT, MRGDAT, IRET)

Input parameters:

FILNAM	CHAR*	Sounding file name

Output parameters:

ISNFLN	INTEGER	Sounding file number
IFLSRC	INTEGER	Data source
NPARM	INTEGER	Number of parameters
PARMS (NPARM)	CHAR*4	Parameter names
IVERT	INTEGER	Vertical coordinate
MRGDAT	LOGICAL	Merged data flag
IRET	INTEGER	Return code 0 = normal return -2 = file could not be opened -7 = file not sounding file -24 = file name is blank

24.18 SN_QDAT - CHECK FOR DATA

This subroutine sets a flag indicating whether data for the current station and time are stored in a file. If the data are not merged, only the mandatory below and above 100 mb and the significant temperature below 100 mb data are checked.

SN_QDAT (ISNFLN, DATFLG, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number

Output parameters:

DATFLG	LOGICAL	Data present flag
IRET	INTEGER	Return code 0 = normal return -4 = file not open -8 = location not set

24.19 SN_QSTN - GET STATION INFORMATION

This subroutine gets station information for the current station. Both the time and station must be set before this subroutine is called.

```
SN_QSTN  ( ISNFLN, STID, ISTNM, SLAT, SLON, SELV, STAT, COUN,
           IRET )
```

Input parameters:

ISNFLN	INTEGER	Sounding file number

Output parameters:

STID	CHAR*4	Station identifier
ISTNM	INTEGER	Station number
SLAT	REAL	Station latitude
SLON	REAL	Station longitude
SELV	REAL	Station elevation
STAT	CHAR*2	State
COUN	CHAR*2	Country
IRET	INTEGER	Return code 0 = normal return -4 = file not open -8 = location not set

24.20 SN_RDAT - READ DATA

This subroutine reads data from a sounding data file. The time and station must be set before calling this subroutine.

SN_RDAT (ISNFLN, NLEV, DATA, IHHMM, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number

Output parameters:

NLEV	INTEGER	Number of levels
DATA (*)	REAL	Station data
IHHMM	INTEGER	Station hour and minute
IRET	INTEGER	Return code 1 = no data at station 0 = normal return -4 = file not open -8 = location not set

24.21 SN_RPRT - READ PART FROM UNMERGED FILE

This subroutine reads data from a sounding data file. This subroutine will only read data from an unmerged data file. The valid part names are: TTAA, TTBB, PPBB, TTCC, TTDD and PPDD. The flag, ZWIND, is used only for significant wind data (PPBB or PPDD). If set, the winds are reported on height surfaces; otherwise, the report is on pressure surfaces.

SN_WPRT (ISNFLN, PART, IHHMM, NLEV, DATA, ZWIND, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
PART	CHAR*4	Part name

Output parameters:

IHHMM	INTEGER	Station time (HHMM)
NLEV	INTEGER	Number of levels
DATA (*)	REAL	Sounding data array
ZWIND	LOGICAL	Flag for sig wind in z coord
IRET	INTEGER	Return code 0 = normal return -4 = file not open -8 = station not set -13 = DM error -17 = invalid merge type -22 = invalid part name

24.22 SN_RTYP - GET LEVEL TYPES

This subroutine returns the report type for each level in a sounding. IDTYPE will be set to 1, 2, or 3 for mandatory, significant temperature or significant wind data. If the data set contains merged data, all the data flags will be set to 1.

SN_RTYP (ISNFLN, NLEV, IDTYPE, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number

Output parameters:

NLEV	INTEGER	Number of levels
IDTYPE (NLEV)	INTEGER	Report type flags 1 = mandatory 2 = sig temperature 3 = sig wind
IRET	INTEGER	Return code 0 = normal return

24.23 SN_SNXT - GET NEXT STATION

This subroutine selects the next station in a sounding file. SN_STIM must be called to set the time before this subroutine is called. Stations to be found may be set in LC_SARE or LC_UARE. Data for this station may be returned or written by calling SN_RDAT or SN_WDAT, respectively.

SN_SNXT (ISNFLN, STID, ISTNM, SLAT, SLON, SELV, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number

Output parameters:

STID	CHAR*4	Station identifier
ISTNM	INTEGER	Station number
SLAT	REAL	Station latitude
SLON	REAL	Station longitude
SELV	REAL	Station elevation
IRET	INTEGER	Return code 0 = normal return -4 = file not open -9 = no more stations -19 = time not set

24.24 SN_SSTN - SET PARTICULAR STATION

This subroutine selects a station in a sounding file. SN_STIM must be called before this subroutine is called. This subroutine will delete any searches set by LC_SARE. Data for this station can be returned or written by calling SN_RDAT or SN_WDAT, respectively.

SN_SSTN (ISNFLN, STN, STID, ISTNM, SLAT, SLON, SELV, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
STN	CHAR*	Station id or number

Output parameters:

STID	CHAR*4	Station identifier
ISTNM	INTEGER	Station number
SLAT	REAL	Station latitude
SLON	REAL	Station longitude
SELV	REAL	Station elevation
IRET	INTEGER	Return code 0 = normal return -4 = file not open -11 = station not in file -19 = time not set

24.25 SN_STIM - SET TIME FOR STATION SEARCH

This subroutine sets the time in a sounding file. All later station searches will return stations corresponding to this time.

SN_STIM (ISNFLN, DATTIM, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
DATTIM	CHAR*	GEMPAK date/time

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -12 = time not in file

24.26 SN_STNF - ADD STATIONS FROM TABLE FILE

This subroutine adds stations from a table file to a sounding file. This subroutine can only be used if the times and stations are not mixed in row and column headers.

SN_STNF (ISNFLN, TBFILE, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
TBFILE	CHAR*	Station table file name

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -6 = too many stations -18 = station file not opened -21 = non-standard file

24.27 SN_TNXT - GET NEXT STATION

This subroutine selects the next time in a sounding file. SN_TSTN must be called to set the station before this subroutine is called. The times will be returned in the order in which they appear in the file rather than in chronological order. Data for this time may be returned or written by calling SN_RDAT or SN_WDAT, respectively.

SN_TNXT (ISNFLN, DATTIM, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number

Output parameters:

DATTIM	CHAR*	GEMPAK date/time
IRET	INTEGER	Return code 0 = normal return -4 = file not open -10 = no more times -20 = station not set

24.28 SN_TSTN - SET STATION SEARCH

This subroutine sets the station in a sounding file. All later time searches will return times corresponding to this station.

SN_TSTN (ISNFLN, STN, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
STN	CHAR*	Station number or id

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -11 = station not in file

24.29 SN_TTIM - SET PARTICULAR TIME

This subroutine sets the time in a sounding file. SN_TSTN must be called before this subroutine is called. Data for this time can be returned or written by calling SN_RDAT or SN_WDAT, respectively.

SN_TTIM (ISNFLN, DATTIM, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
DATTIM	CHAR*	GEMPAK date/time

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -12 = time not found -20 = station not set

24.30 SN_USTN - UPDATE STATION INFORMATION

This subroutine updates the header information for a station in a sounding data file. This subroutine can only be used if the times and stations are not mixed in row or column headers.

```
SN_USTN ( ISNFLN, STID, ISTNM, SLAT, SLON, SELV, STAT, COUN,
          KEYNAM, IRET )
```

Input parameters:

ISNFLN	INTEGER	Sounding file number
STID	CHAR*	Station number or id
ISTNM	INTEGER	Station number
SLAT	REAL	Station latitude
SLON	REAL	Station longitude
SELV	REAL	Station elevation
STAT	CHAR*2	State
COUN	CHAR*2	Country
KEYNAM	CHAR*4	Key to update (STID or STNM)

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -11 = station not in file -13 = DM error -21 = non-standard file

24.31 SN_WDAT - WRITE DATA

This subroutine writes data to a sounding data file. The time and station must both be set before this subroutine is called. The station time will be stored if the station time flag, STMFLG, was set when the file was created. This subroutine will only write data to a merged data file. The subroutine SN_WPRT must be used to write data to an unmerged file.

SN_WDAT (ISNFLN, IHHMM, NLEV, DATA, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
IHHMM	INTEGER	Station time (HHMM)
NLEV	INTEGER	Number of levels
DATA (*)	REAL	Sounding data array

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -8 = station not set -13 = DM error -17 = invalid merge type

24.32 SN_WPRT - WRITE PART TO UNMERGED FILE

This subroutine writes data to an unmerged sounding data file. The time and station must both be set before this subroutine is called. The station time will be stored if the station time flag, STMFLG, was set when the file was created. This subroutine will only write data to an unmerged data file. The subroutine SN_WDAT must be used to write data to a merged file. The valid part names are: TTAA, TTBB, PPBB, TTCC, TTDD and PPDD. The flag, ZWIND, is used only for significant wind data (PPBB or PPDD). If set, the winds are reported on height surfaces; otherwise, the report is on pressure surfaces.

SN_WPRT (ISNFLN, PART, IHHMM, NLEV, DATA, ZWIND, IRET)

Input parameters:

ISNFLN	INTEGER	Sounding file number
PART	CHAR*4	Part name
IHHMM	INTEGER	Station time (HHMM)
NLEV	INTEGER	Number of levels
DATA (*)	REAL	Sounding data array
ZWIND	LOGICAL	Flag for sig wind in z coord

Output parameters:

IRET	INTEGER	Return code 0 = normal return -4 = file not open -8 = station not set -13 = DM error -17 = invalid merge type -22 = invalid part name

CHAPTER 25

SYSTEM SERVICES (SS) LIBRARY

SS_CURS	Move cursor to specified line
SS_EXIT	Terminate a program
SS_GSYM	Get symbol value
SS_GTIM	Get the system time
SS_IRET	Set I/O return codes
SS_PAGE	Clear terminal screen
SS_PATH	Change pathname for file
SS_WAIT	Halt program for specified time

System Services (SS) Library Summary

The system services library contains machine-dependent system-service calls in order to isolate them for conversion to other machines.

The routines SS_GLUN and SS_FLUN have been replaced by FORTRAN standard routines, FL_GLUN and FL_FLUN.

SS_IRET translates the IOSTAT return from a FORTRAN I/O call into a GEMPAK FL error number.

SS_GSYM returns the value of a VMS symbol. It is only used in the TAE (IP) library to determine whether the user is in the TAE or not.

SS_GTIM returns the current system clock time in GEMPAK format. It is used in real-time data decoders to complete the bulletin time.

The subroutines SS_CURS and SS_PAGE allow the programmer control over the output display. They are not generally called by GEMPAK programs and can be replaced by subroutines which do nothing. SS_WAIT is included for convenience. The FL routines that use SS_WAIT are likely to need rewriting for non-VMS systems, so it is possible that SS_WAIT also can be replaced by a dummy subroutine in ports to non-VMS systems.

ERROR MESSAGES:

[SS 208]	The logical unit number was not freed.
[SS 204]	There are no more logical unit numbers.
[SS -1]	System service error.

SYSTEM SERVICES (SS) LIBRARY

SS Library Calls

```
SS_CURS   ( line, / iret )
SS_GSYM   ( symnam, / symval, iret )
SS_GTIM   ( / dattim, iret )
SS_IRET   ( iostat, / iflerr, iret )
SS_PAGE   ( / iret )
SS_PATH   ( filnam, path, / newfil, iret )
SS_WAIT   ( nsec, / iret )
```

25.1 SS_CURS - MOVE CURSOR TO SPECIFIED LINE

This subroutine moves the cursor to the beginning of the specified line.

SS_CURS (LINE, IRET)

Input parameters:

LINE	INTEGER	Line on which to put cursor

Output parameters:

IRET	INTEGER	Return code 0 = normal return -1 = system service error

25.2 SS_EXIT - TERMINATE A PROGRAM

This subroutine terminates program execution. It has no input or output parameters.

SS_EXIT

25.3 SS_GSYM - GET SYMBOL VALUE

This subroutine gets the value of a symbol defined on a VMS system.

SS_GSYM (SYMNAM, SYMVAL, IRET)

Input parameters:

SYMNAM	CHAR*	Symbol name

Output parameters:

SYMVAL	CHAR*	Symbol value
IRET	INTEGER	Return code 0 = normal return -1 = symbol not found

25.4 SS_GTIM - GET THE SYSTEM TIME

This subroutine returns the current system clock time as a GEMPAK date/time.

SS_GTIM (DATTIM, IRET)

Output parameters:

DATTIM	CHAR*	System time in GEMPAK format
IRET	INTEGER	Return code 0 = normal return -1 = system service error

25.5 SS_IRET - SET I/O RETURN CODES

This subroutine takes the IOSTAT value returned from a FORTRAN I/O statement and determines the GEMPAK message number for the error. GEMPAK routines expect IFLERR = 0 for a normal return.

SS_IRET (IOSTAT, IFLERR, IRET)

Input parameters:

IOSTAT	INTEGER	Status from I/O operation

Output parameters:

IFLERR	INTEGER	GEMPAK file error
IRET	INTEGER	Return code 0 = normal return

25.6 SS_PAGE - CLEAR TERMINAL SCREEN

This subroutine clears the terminal screen.

SS_PAGE (IRET)

Output parameters:

IRET	INTEGER	Return code 0 = normal return

25.7 SS_PATH - CHANGE PATHNAME FOR FILE

This subroutine takes an input file and path name and appends the actual file to the path. If a path was specified in the input file, it is replaced by PATH.

SS_PATH (FILNAM, PATH, NEWFIL, IRET)

Input parameters:

FILNAM	CHAR*	Input file name
PATH	CHAR*	Path name for output file

Output parameters:

NEWFIL	CHAR*	Output file name
IRET	INTEGER	Return code 0 = normal return

25.8 SS_WAIT - HALT PROGRAM FOR SPECIFIED TIME

This subroutine halts the execution of a program for up to 420 seconds (7 minutes).

SS_WAIT (NSEC, IRET)

Input parameters:

NSEC	INTEGER	Number of seconds to wait

Output parameters:

IRET	INTEGER	Return code 0 = normal return

CHAPTER 26

STRING (ST) LIBRARY

ST_ABBR	Check for abbreviation
ST_ALNM	Check for alphanumeric character
ST_C2C	Convert string to character array
ST_C2I	Convert string to integer array
ST_C2R	Convert string to real array
ST_CLST	Convert character list to array
ST_CRNM	Decode real
ST_CTOI	Store characters in integers
ST_FIND	Find a string in a list
ST_FWRD	Find Nth word in a string
ST_ILST	Convert integer list to array
ST_INCH	Encode integer
ST_INLN	Encode integer
ST_INTG	Decode integer
ST_ITOC	Recover strings from integers
ST_ITOS	Recover string from integer array
ST_LCUC	Convert string to upper case
ST_LDSP	Remove leading spaces
ST_LSTR	Find length of string
ST_NOCC	Find Nth occurrence of character
ST_NUMB	Decode integer
ST_NXTS	Search string for substring list
ST_RANG	Get range limits
ST_RLCH	Encode real number
ST_RLST	Convert real list to array
ST_RMBL	Remove blanks
ST_RMST	Find a substring within a string
ST_RNAN	Remove non-alphanumerics
ST_RXBL	Remove extra blanks
ST_STOI	Store string in integer array
ST_UCLC	Convert string to lower case
ST_UNPR	Remove control characters
ST_UTAB	Replace tabs with spaces

String (ST) Library Summary

The GEMPAK string library provides subroutines to simplify handling character strings. These subroutines are used extensively throughout GEMPAK.

Basic routines convert strings to upper or lower case (ST_LCUC and ST_UCLC), determine the length without trailing blanks (ST_LSTR), remove leading spaces (ST_LDSP), and check for alphanumeric characters (ST_ALNM).

The routines ST_C2C, ST_C2I, and ST_C2R separate a character string into arrays of characters, integers or reals. Separators are any non-alphanumeric character, except period, plus, minus, or asterisk.

The routines ST_CLST, ST_ILST, and ST_RLST also separate lists into arrays. In this case, the separator must be specified (and may not be a blank) and a default value is inserted for unspecified values. These subroutines are especially useful for decoding GEMPAK input parameters and are preferred to the subroutines described in the last paragraph.

The routines ST_INCH, ST_RLCH, ST_NUMB, and ST_CRNM encode and decode integers and real numbers.

ERROR MESSAGES:

[ST 1] More than the expected number of values entered.
[ST -1] Invalid input string.
[ST -2] Conversion error.
[ST -3] Substring not found or invalid.
[ST -4] Word not found.
[ST -5] Nth occurrence not found.

STRING (ST) LIBRARY

ST Library Calls

ST_ABBR	(string, stabbr, / abbr, iret)
ST_ALNM	(chrstr, / ityp, iret)
ST_C2C	(string, nexp, / strarr, num, iret)
ST_C2I	(string, nexp, / intarr, num, iret)
ST_C2R	(string, nexp, / rarr, num, iret)
ST_CLST	(string, sep, cdef, nexp, / carr, num, iret)
ST_CRNM	(string, / value, iret)
ST_CTOI	(carray, nval, / iarray, iret)
ST_FIND	(string, stlist, nstr, / ipos, iret)
ST_FWRD	(string, ifirst, ilast, nword, / istrt, iend, iret)
ST_ILST	(string, sep, idef, nexp, / iarr, num, iret)
ST_INCH	(intg, / string, iret)
ST_INLN	(intg, / string, lens, iret)
ST_INTG	(string, / intg, iret)
ST_ITOC	(iarray, nval, / carray, iret)
ST_ITOS	(iarray, nval, / nchar, string, iret)
ST_LCUC	(string, / outstr, iret)
ST_LDSP	(string, / outstr, ncout, iret)
ST_LSTR	(string, / lens, iret)
ST_NOCC	(string, chocc, nocc, / ipoint, iret)
ST_NUMB	(string, / ival, iret)
ST_NXTS	(string, ifirst, ilast, stlist, ilens, nstr, / ipos, istrg, iret)
ST_RANG	(string, / first, last, inc, itype, iret)
ST_RLCH	(rlnum, np, / string, iret)

ST_RLST (string, sep, rdef, nexp, / rarr, num, iret)

ST_RMBL (string, / outstr, length, iret)

ST_RMST (string, substr, / ipos, outstr, iret)

ST_RNAN (string, / outstr, length, iret)

ST_RXBL (string, / outstr, length, iret)

ST_STOI (string, nchar, / nval, iarray, iret)

ST_UCLC (string, / outstr, iret)

ST_UNPR (string, lenin, / outstr, lenout, iret)

ST_UTAB (string, nchar, / outstr, iret)

STRING (ST) LIBRARY

26.1 ST_ABBR - CHECK FOR ABBREVIATION

This subroutine determines whether the string in STABBR is an abbreviation (beginning substring) of STRING. Both strings are converted to upper case before the comparison is done.

ST_ABBR (STRING, STABBR, ABBR, IRET)

Input parameters:

STRING	CHAR*	Full string
STABBR	CHAR*	Abbreviation

Output parameters:

ABBR	LOGICAL	Abbreviation flag
IRET	INTEGER	Return code 0 = normal return

26.2 ST_ALNM - CHECK FOR ALPHANUMERIC CHARACTER

This subroutine determines whether a character is a letter, number or non-alphanumeric character.

ST_ALNM (CHRSTR, ITYP, IRET)

Input parameters:

CHRSTR	CHAR*1	Character to analyze

Output parameters:

ITYP	INTEGER	Character type 0 = non-alphanumeric 1 = number 2 = letter
IRET	INTEGER	Return code 0 = normal return

26.3 ST_C2C - CONVERT STRING TO CHARACTER ARRAY

This subroutine breaks a string to an array of strings. The string separators may be any non-alphanumeric character except a minus sign (-), plus sign (+), asterisk (*) or period (.).

ST_C2C (STRING, NEXP, STRARR, NUM, IRET)

Input parameters:

STRING	CHAR*	String
NEXP	INTEGER	Maximum number of strings

Output parameters:

STRARR (*)	CHAR*	String array
NUM	INTEGER	Number of strings
IRET	INTEGER	Return code 1 = more than NEXP strings 0 = normal return -1 = no strings input

26.4 ST_C2I - CONVERT STRING TO INTEGER ARRAY

This subroutine breaks a string to an array of integers. The integers may be separated by any non-alphanumeric character except a minus sign (-), a plus sign (+), a period (.) or an asterisk (*).

ST_C2I (STRING, NEXP, INTARR, NUM, IRET)

Input parameters:

STRING	CHAR*	String
NEXP	INTEGER	Maximum number of integers

Output parameters:

INTARR (*)	INTEGER	Integer array
NUM	INTEGER	Number of integers
IRET	INTEGER	Return code 1 = more than NEXP integers 0 = normal return -1 = invalid string -2 = conversion error

26.5 ST_C2R - CONVERT STRING TO REAL ARRAY

This subroutine converts a string into an array of real numbers. The numbers may be separated by any non-alphanumeric character except a period (.), a plus sign (+), a minus sign (-) or an asterisk (*).

ST_C2R (STRING, NEXP, RARR, NUM, IRET)

Input parameters:

STRING	CHAR*	String
NEXP	INTEGER	Maximum number of reals

Output parameters:

RARR (NUM)	INTEGER	Converted real array
NUM	INTEGER	Number of converted reals
IRET	INTEGER	Return code 1 = more than NEXP reals 0 = normal return -1 = invalid string -2 = conversion error

26.6 ST_CLST - CONVERT CHARACTER LIST TO ARRAY

This subroutine breaks a string containing a list of strings into an array of strings. The separator for the strings is input as SEP. If the separator is a blank, multiple blanks will be changed to single blanks before the string is processed. If null strings are encountered or fewer than NEXP strings are found in the string, the appropriate CARR locations are set to CDEF.

ST_CLST (STRING, SEP, CDEF, NEXP, CARR, NUM, IRET)

Input parameters:

STRING	CHAR*	String
SEP	CHAR*1	Separator
CDEF	CHAR*	Default string
NEXP	INTEGER	Number of expected values

Output parameters:

CARR (NUM)	CHAR*	Array of strings
NUM	INTEGER	Number of strings returned
IRET	INTEGER	Return code 1 = more than NEXP values 0 = normal return

26.7 ST_CRNM - DECODE REAL

This subroutine converts a character string to a real number. If the conversion fails, RMISSD is returned.

ST_CRNM (STRING, VALUE, IRET)

Input parameters:

STRING	CHAR*	String

Output parameters:

VALUE	REAL	Real number
IRET	INTEGER	Return code 0 = normal return -2 = conversion error

26.8 ST_CTOI - STORE CHARACTERS IN INTEGERS

This subroutine stores an array of 4-character strings in an array of integers. Each integer element contains one of the 4-character strings.

ST_CTOI (CARRAY, NVAL, IARRAY, IRET)

Input parameters:

CARRAY (NVAL)	CHAR*4	Character array
NVAL	INTEGER	Number of strings

Output parameters:

IARRAY (NVAL)	INTEGER	Integer array
IRET	INTEGER	Return code 0 = normal return -2 = conversion error

26.9 ST_FIND - FIND A STRING IN A LIST

This subroutine searches for a particular string in a list of strings. The position in the array is returned in IPOS. If the string is not found, IPOS is set to 0.

ST_FIND (STRING, STLIST, NSTR, IPOS, IRET)

Input parameters:

STRING	CHAR*	String
STLIST (NSTR)	CHAR*	List of strings
NSTR	INTEGER	Number of strings in list

Output parameters:

IPOS	INTEGER	Position of string in list 0 = not found
IRET	INTEGER	Return code 0 = normal return

26.10 ST_FWRD - FIND NTH WORD IN A STRING

This subroutine returns pointers to the word which is NWORDs after the IFIRST character in the string. Words are assumed to be delimited by blanks.

ST_FWRD (STRING, IFIRST, ILAST, NWORD, ISTRT, IEND, IRET)

Input parameters:

STRING	CHAR*	String
IFIRST	INTEGER	First character to check
ILAST	INTEGER	Last character to check
NWORD	INTEGER	Word number

Output parameters:

ISTRT	INTEGER	Pointer to start of word
IEND	INTEGER	Pointer to end of word
IRET	INTEGER	Return code 0 = normal return -4 = word not found

26.11 ST_ILST - CONVERT INTEGER LIST TO ARRAY

This subroutine breaks a string containing a list of integers into an array of integers. The separator for the integers is input as SEP. If the separator is a blank, multiple blanks will be changed to single blanks before the string is processed. If null strings are encountered or fewer than NEXP strings are found in the string, the appropriate IARR locations are set to IDEF.

ST_ILST (STRING, SEP, IDEF, NEXP, IARR, NUM, IRET)

Input parameters:

STRING	CHAR*	String
SEP	CHAR*1	Separator
IDEF	INTEGER	Default value
NEXP	INTEGER	Number of expected values

Output parameters:

IARR (NUM)	INTEGER	Array of integer values
NUM	INTEGER	Number of values returned
IRET	INTEGER	Return code 1 = more than NEXP values 0 = normal return -3 = invalid substring

26.12 ST_INCH - ENCODE INTEGER

This subroutine encodes an integer in a character string.

ST_INCH (INTG, STRING, IRET)

Input parameters:

INTG	INTEGER	Integer

Output parameters:

STRING	CHAR*	Encoded value
IRET	INTEGER	Return code 0 = normal return -2 = error on conversion

26.13 ST_INLN - ENCODE INTEGER

This subroutine converts an integer to a character string. Unlike ST_INCH, the length of the string is returned.

ST_INLN (INTG, STRING, LENS, IRET)

Input parameters:

INTG	INTEGER	Integer

Output parameters:

STRING	CHAR*	String
LENS	INTEGER	Length of string
IRET	INTEGER	Return code 0 = normal return -2 = conversion error

26.14 ST_INTG - DECODE INTEGER

This subroutine decodes a character string into an integer. If the string cannot be decoded, INTG is set to IMISSD. Note that only the substring containing the digits to be decoded should be sent to this subroutine, rather than a string with trailing blanks.

ST_INTG (STRING, INTG, IRET)

Input parameters:

STRING	CHAR*	Input string

Output parameters:

INTG	INTEGER	Decoded integer
IRET	INTEGER	Return code 0 = normal return -2 = conversion error

26.15 ST_ITOC - RECOVER STRINGS FROM INTEGERS

This subroutine decodes an array of integers containing four characters each into a character string array.

ST_ITOC (IARRAY, NVAL, CARRAY, IRET)

Input parameters:

IARRAY (NVAL)	INTEGER	Integer array
NVAL	INTEGER	Number of integers

Output parameters:

CARRAY (NVAL)	CHAR*4	Character array
IRET	INTEGER	Return code 0 = normal return -2 = conversion error

26.16 ST_ITOS - RECOVER STRING FROM INTEGER ARRAY

This subroutine decodes an array of integers which contain four characters each into a single character string.

ST_ITOS (IARRAY, NVAL, NCHAR, STRING, IRET)

Input parameters:

IARRAY (NVAL)	INTEGER	Integer array
NVAL	INTEGER	Number of integers

Output parameters:

NCHAR	INTEGER	Number of characters
STRING	CHAR*	Character string
IRET	INTEGER	Return code 0 = normal return -2 = conversion error

26.17 ST_LCUC - CONVERT STRING TO UPPER CASE

This subroutine converts lower-case characters in a string to upper case. The input and output string may be the same variable.

ST_LCUC (STRING, OUTSTR, IRET)

Input parameters:

STRING	CHAR*	String

Output parameters:

OUTSTR	CHAR*	String in upper case
IRET	INTEGER	Return code 0 = normal return

26.18 ST_LDSP - REMOVE LEADING SPACES

This subroutine deletes the leading spaces and tabs in a string. The input and output strings may be the same variable.

ST_LDSP (STRING, OUTSTR, NCOUT, IRET)

Input parameters:

STRING	CHAR*	String

Output parameters:

OUTSTR	CHAR*	Output string
NCOUT	INTEGER	Number of characters output
IRET	INTEGER	Return code 0 = normal return

26.19 ST_LSTR - FIND LENGTH OF STRING

This subroutine returns the number of characters in a string disregarding trailing null characters, tabs and spaces.

ST_LSTR (STRING, LENS, IRET)

Input parameters:		
STRING	CHAR*	String
Output parameters:		
LENS	INTEGER	Length of string
IRET	INTEGER	Return code 0 = normal return

26.20 ST_NOCC - FIND NTH OCCURRENCE OF CHARACTER

This subroutine finds the Nth occurrence of a character in a string.

ST_NOCC (STRING, CHOCC, NOCC, IPOINT, IRET)

Input parameters:

STRING	CHAR*	String
CHOCC	CHAR*	Search character
NOCC	INTEGER	Occurrence to find

Output parameters:

IPOINT	INTEGER	Pointer to Nth occurrence
IRET	INTEGER	Return code 0 = normal return -5 = Nth occurrence not found

26.21 ST_NUMB - DECODE INTEGER

This subroutine converts a string into an integer.

ST_NUMB (STRING, IVAL, IRET)

Input parameters:

STRING	CHAR*	String

Output parameters:

IVAL	INTEGER	Integer value
IRET	INTEGER	Return code 0 = normal return -2 = conversion error

26.22 ST_NXTS - SEARCH STRING FOR SUBSTRING LIST

This subroutine returns a pointer to the first occurrence of any of a list of substrings within a given string.

ST_NXTS (STRING, IFIRST, ILAST, STLIST, ILENS, NSTR, IPOS, ISTRG, IRET)

Input parameters:

STRING	CHAR*	Input string
IFIRST	INTEGER	First position to check
ILAST	INTEGER	Last position to check
STLIST (NSTR)	CHAR*	List of substrings
ILENS (NSTR)	INTEGER	Lengths of substrings
NSTR	INTEGER	Number of substrings

Output parameters:

IPOS	INTEGER	Position of first substring
ISTRG	INTEGER	Array element number of string
IRET	INTEGER	Return code 0 = normal return -3 = substring not found

26.23 ST_RANG - GET RANGE LIMITS

This subroutine changes a string range into the beginning, end, and increment values. The values must be separated by '-'.

ST_RANG (STRING, FIRST, LAST, INC, ITYPE, IRET)

Input parameters:

STRING	CHAR*	String

Output parameters:

FIRST	CHAR*	First value in range
LAST	CHAR*	Last value in range
INC	CHAR*	Range increment
ITYPE	INTEGER	Range type 0 = no range input 1 = range without increment 2 = range with increment
IRET	INTEGER	Return code 0 = normal return

26.24 ST_RLCH - ENCODE REAL NUMBER

This subroutine encodes a real number in a character string. NP contains the number of decimal places to be included in the output string. RLNUM is rounded to NP decimal places.

ST_RLCH (RLNUM, NP, STRING, IRET)

Input parameters:

RLNUM	REAL	Real number
NP	INTEGER	Number of decimal places

Output parameters:

STRING	CHAR*	Output string
IRET	INTEGER	Return code
		0 = normal return

26.25 ST_RLST - CONVERT REAL LIST TO ARRAY

This subroutine breaks a string containing a list of reals into an array of real values. The separator for the reals is input as SEP. If the separator is a blank, multiple blanks will be changed to single blanks before the string is processed. If null strings are encountered or fewer than NEXP strings are found in the string, the appropriate RARR locations are set to RDEF.

ST_RLST (STRING, SEP, RDEF, NEXP, RARR, NUM, IRET)

Input parameters:

STRING	CHAR*	String
SEP	CHAR*1	Separator
RDEF	REAL	Default value
NEXP	INTEGER	Number of expected values

Output parameters:

RARR (NUM)	REAL	Array of real values
NUM	INTEGER	Number of values returned
IRET	INTEGER	Return code 1 = too many values 0 = normal return -3 = invalid substring

26.26 ST_RMBL - REMOVE BLANKS

This subroutine removes spaces and tabs from a string. The input and output strings may be the same variable.

ST_RMBL (STRING, OUTSTR, LENGTH, IRET)

Input parameters:

STRING	CHAR*	String

Output parameters:

OUTSTR	CHAR*	String without blanks
LENGTH	INTEGER	Length of output string
IRET	INTEGER	Return code 0 = normal return

26.27 ST_RMST - FIND A SUBSTRING WITHIN A STRING

This subroutine finds a substring within a string and returns the position of that substring and the output string with the substring removed. If the substring is not found, the position, IPOS, is set to zero.

ST_RMST (STRING, SUBSTR, IPOS, OUTSTR, IRET)

Input parameters:

STRING	CHAR*	String
SUBSTR	CHAR*	Substring

Output parameters:

IPOS	INTEGER	Position of substring
OUTSTR	CHAR*	Output string less substring
IRET	INTEGER	Return code 0 = normal return

26.28 ST_RNAN - REMOVE NON-ALPHANUMERICS

This subroutine replaces non-alphanumeric characters with spaces and removes the extra spaces from a character string. The characters period (.), plus sign (+), minus sign (-) and asterisk (*) are not removed.

ST_RNAN (STRING, OUTSTR, LENGTH, IRET)

Input parameters:

STRING	CHAR*	String

Output parameters:

OUTSTR	CHAR*	Converted string
LENGTH	INTEGER	Length of output string
IRET	INTEGER	Return code
		0 = normal return

26.29 ST_RXBL - REMOVE EXTRA BLANKS

This subroutine removes extra spaces and tabs from a string. Only single blanks will separate substrings. The input and output strings may be the same variable.

ST_RXBL (STRING, OUTSTR, LENGTH, IRET)

Input parameters:

STRING	CHAR*	String

Output parameters:

OUTSTR	CHAR*	String without blanks
LENGTH	INTEGER	Length of output string
IRET	INTEGER	Return code 0 = normal return

26.30 ST_STOI - STORE STRING IN INTEGER ARRAY

This subroutine stores a character string in an integer array. Four characters are written to each integer.

ST_STOI (STRING, NCHAR, NVAL, IARRAY, IRET)

Input parameters:

STRING	CHAR*	String
NCHAR	INTEGER	Number of characters to store

Output parameters:

NVAL	INTEGER	Number of integers
IARRAY (NVAL)	INTEGER	Integer array
IRET	INTEGER	Return code 0 = normal return -2 = conversion error

26.31 ST_UCLC - CONVERT STRING TO LOWER CASE

This subroutine converts upper-case characters in a string to lower case. The input and output strings may be the same variable.

ST_UCLC (STRING, OUTSTR, IRET)

Input parameters:		
STRING	CHAR*	String
Output parameters:		
OUTSTR	CHAR*	String in upper case
IRET	INTEGER	Return code
		0 = normal return

26.32 ST_UNPR - REMOVE CONTROL CHARACTERS

This subroutine eliminates substrings of unprintable characters. Substrings of control characters, i.e., characters less than a blank, are replaced by a single blank. Characters greater than '}' (CHAR (126)) are replaced by '~' (CHAR (127)). This subroutine can be used to replace control characters such as CR and LF with a single blank. Invalid characters in the ASCII character set are replaced by '~' so that the lengths of fields in the record will remain unchanged. The input and output strings may be the same variable.

ST_UNPR (STRING, LENIN, OUTSTR, LENOUT, IRET)

Input parameters:

STRING	CHAR*	Input string
LENIN	INTEGER	Length of input string

Output parameters:

OUTSTR	CHAR*	Output string
LENOUT	INTEGER	Length of output string
IRET	INTEGER	Return code 0 = normal return

26.33 ST_UTAB - REPLACE TABS WITH SPACES

This subroutine substitutes spaces for tabs in a string. Spaces are added for each tab found so that the character after the tab appears at the next tab stop. Tab stops are assumed to be at positions 9, 17, 25, ... The input and output strings may be the same variable.

ST_UTAB (STRING, NCHAR, OUTSTR, IRET)

Input parameters:

STRING	CHAR*	Input string
NCHAR	INTEGER	Number of characters

Output parameters:

OUTSTR	CHAR*	Output string
IRET	INTEGER	Return code 0 = normal return

CHAPTER 27

TABLE (TB) LIBRARY

TB_FGEO	Find geographic area
TB_GRNV	Read grid navigation table
TB_PCNV	Read parameter conversion table
TB_PRMT	Read parameter flag table
TB_RSTN	Read station table
TB_WGEO	Write to geographic table

TABLE (TB) LIBRARY

Table (TB) Library Summary

The table library contains subroutines to access GEMPAK table files. A table file is a sequential file which may have leading comment records. A comment record is any record where the first non-blank character is an exclamation point. Table files may be created using a text editor.

The subroutine FL_TOPN may be used to open any table file for read access. The file will be positioned after the last comment record. FL_SWOP will open a table file for write access. FL_APND will position the file after the last record if information is to be added to the file.

The following paragraphs describe the current GEMPAK tables which are located in GEMTABL.

GEOGRAPHIC TABLE

The geographic table contains a list of geographic abbreviations, a full geographic name, the center latitude and longitude, and a latitude and longitude range. The subroutine TB_FGEO will search the table for a geographic abbreviation. The GEMPAK table GEOG.TBL will be searched unless the first character in the requested name is a #, in which case the local file GEOG.TBL will be used. TB_WGEO will write a new geographic area to the file. The format statement used to read or write to the file is: (A8, A18, 4F8.2).

GRID NAVIGATION TABLE

The grid navigation table associates a 4-character name with a 3-digit identification number and a complete set of grid navigation parameters (projection name, 3 projection angles, 4 bounding lat/lon values, number of x and y grid points). Grid numbers less than 300 correspond to standard NMC grid numbers. The table also contains the suggested DELTAN and EXTEND parameters for performing a Barnes analysis. The subroutine TB_GRNV can be used to read the table; entries are free-format, but all 14 parameters must be non-blank.

PARAMETER CONVERSION TABLE and PARAMETER FLAG TABLE

These tables are used by the PC library to compute requested output parameters from the parameters contained in a data set. The subroutines TB_PCNV and TB_PRMT can be used to read the tables. The GEMPAK tables are named PCCONV.TBL and PRMFLG.TBL. In general, these files may be modified to add new parameters, but the subroutines to read the files will be called only by the PC library subroutines.

TABLE (TB) LIBRARY

SURFACE STATION TABLE, UPPER-AIR STATION TABLE, and WORLD UA STATION TABLE

These tables contain surface and upper-air stations for the United States, Canada, Mexico and the Caribbean, which report on the Domestic Data line and upper-air data stations for the world. The GEMPAK tables are SFSTNS.TBL, SNSTNS.TBL and SNWORLD.TBL. Note that SFSTNS.TBL was previously named STATIONS.TBL. TB_RSTN will read a single record from the station file. Each record of the file contains the station identifier, number, name, state, country, latitude, longitude and elevation, and can be read with the format: (A4, 1X, I6, 1X, A32, 1X, A2, 1X, A2, 1X, I5, 1X, I6, 1X, I5).

SURFACE DATA PACKING TABLE and UPPER-AIR DATA PACKING TABLE

These packing tables contain recommended parameters and packing information for data received from the 604-line or Domestic Data Service.

The current GEMPAK surface packing file is SFPACK.TBL. The file is read by DP_FILE, which is called by SF_CRFP when a packed surface file is created.

The current GEMPAK upper-air packing file is SNPACK.TBL . Note that the upper-air significant and mandatory reports are usually stored separately in unmerged upper-air files. The subroutine SN_CRUA, which creates such a file, may specify whether the data are to be packed, but the packing information is predetermined.

ERROR MESSAGES:

[TB -1] End of file reached.
[TB -2] Read error.
[TB -3] The table ... cannot be opened.
[TB -4] The geographic area ... is not in the table.
[TB -5] Error writing to the file.
[TB -6] The parameter conversion buffer is full.
[TB -7] The parameter flag buffer is full.
[TB -9] Error converting table parameters.

TB Library Calls

TB_FGEO	(geog, / cenlat, cenlon, diflat, diflon, iret)
TB_GRNV	(lun, / namgd, numgd, prjgd, anggd, gargd, nxgd, nygd, deln, extnd, iret)
TB_PCNV	(maxfnc, / nfunc, parms, funcs, prmin, iret)
TB_PRMT	(maxprm, / nparms, parms, chrflg, intflg, extflg, angflg, iret)
TB_RSTN	(lun, / stid, stnnam, istnm, stat, coun, slat, slon, selv, iret)
TB_WGEO	(lun, geoare, geonam, cenlat, cenlon, diflat, diflon, / iret)

27.1 TB_FGEO - FIND GEOGRAPHIC AREA

This subroutine searches the geographic name table for an area and returns the center latitude and longitude and the latitude and longitude range. If the first character in GEOG is #, the rest of the name is used in searching the user's geographic table, GEOG.TBL.

If the geographic table cannot be opened, an error message is written.

The input parameter GEOG must be in upper case.

TB_FGEO (GEOG, CENLAT, CENLON, DIFLAT, DIFLON, IRET)

Input parameters:

GEOG	CHAR*	Geographic name

Output parameters:

CENLAT	REAL	Center latitude
CENLON	REAL	Center longitude
DIFLAT	REAL	Latitude range of area
DIFLON	REAL	Longitude range of area
IRET	INTEGER	Return code 0 = normal return -3 = table not opened -4 = area not in table

27.2 TB_GRNV - READ GRID NAVIGATION TABLE

This subroutine returns the contents of a line in a GEMPAK grid navigation table. Table entries are free format, but no entry may be blank.

TB_GRNV (LUN, NAMGD, NUMGD, PRJGD, ANGGD, GARGD, NXGD, NYGD, DELN, EXTND, IRET)

Input parameters:

LUN	INTEGER	Logical unit number

Output parameters:

NAMGD	CHAR*4	Grid type name
NUMGD	INTEGER	Grid type number
PRJGD	CHAR*	Projection name
ANGGD (3)	REAL	Grid projection angles
GARGD (4)	REAL	Grid lat/lon corners
NXGD	INTEGER	Number of grid pts in x dir
NYGD	INTEGER	Number of grid pts in y dir
DELN	REAL	DELTA N for Barnes analysis
EXTND	REAL	Grid size increase, first pass
IRET	INTEGER	Return code 0 = normal return -1 = end of file reached -2 = read error -9 = decode error

27.3 TB_PCNV - READ PARAMETER CONVERSION TABLE

This subroutine reads in the parameter-function computation table and decomposes it into functions and required parameters. If the function table cannot be opened or is too large for the buffer space, an error message is written, but no error is returned.

TB_PCNV (MAXFNC, NFUNC, PARMS, FUNCS, PRMIN, IRET)

Input parameters:

MAXFNC	INTEGER	Maximum number of functions

Output parameters:

NFUNC	INTEGER	Number of functions
PARMS (NFUNC)	CHAR*	Computable functions
FUNCS (NFUNC)	CHAR*	Function names
PRMIN (4,NFUNC)	CHAR*	Input parameters to funcs
IRET	INTEGER	Return code 0 = normal return

27.4 TB_PRMT - READ PARAMETER FLAG TABLE

This subroutine reads the parameter type table and returns the parameters and the character, interpolation, extrapolation and angle flags. If the file cannot be opened or there are too many parameters, an error message will be written, but no error will be returned.

```
TB_PRMT  ( MAXPRM, NPARMS, PARMS, CHRFLG, INTFLG, EXTFLG, ANGFLG,
           IRET )
```

Input parameters:

MAXPRM	INTEGER	Maximum number of parameters

Output parameters:

NPARMS	INTEGER	Number of parameters
PARMS (NPARMS)	CHAR*	Parameter names
CHRFLG (NPARMS)	LOGICAL	Character type flags
INTFLG (NPARMS)	LOGICAL	Interpolation flags
EXTFLG (NPARMS)	LOGICAL	Extrapolation flags
ANGFLG (NPARMS)	LOGICAL	Angle flags
IRET	INTEGER	Return code 0 = normal return

27.5 TB_RSTN - READ STATION TABLE

This subroutine reads the next record from the GEMPAK station table.

```
TB_RSTN ( LUN, STID, STNNAM, ISTNM, STAT, COUN, SLAT, SLON,
          SELV, IRET )
```

Input parameters:

LUN	INTEGER	Logical unit number

Output parameters:

STID	CHAR*4	Station identifier
STNNAM	CHAR*32	Station name
ISTNM	INTEGER	Station number
STAT	CHAR*2	State
COUN	CHAR*2	Country
SLAT	REAL	Station latitude
SLON	REAL	Station longitude
SELV	REAL	Station elevation
IRET	INTEGER	Return code 0 = normal return -1 = end of file -2 = read error

27.6 TB_WGEO - WRITE TO GEOGRAPHIC TABLE

This subroutine writes a record to a GEMPAK geographic table file. The file should be positioned at the EOF mark using FL_APND before this subroutine is called. Otherwise, information already in the table will be overwritten.

TB_WGEO (LUN, GEOARE, GEONAM, CENLAT, CENLON, DIFLAT, DIFLON, IRET)

Input parameters:

LUN	INTEGER	Logical unit number
GEOARE	CHAR*8	Area name abbreviation
GEONAM	CHAR*18	Area name
CENLAT	REAL	Center latitude
CENLON	REAL	Center longitude
DIFLAT	REAL	Latitude range
DIFLON	REAL	Longitude range

Output parameters:

IRET	INTEGER	Return code 0 = normal return -5 = write error

CHAPTER 28

GRID TIME (TG) LIBRARY

TG_CFTM	Create character forecast time
TG_CTOI	Grid time to integers
TG_DIFF	Grid time difference
TG_FLST	Find times in file
TG_FTOI	File time to integers
TG_FULL	Partial to full grid time
TG_IFTM	Create integer forecast time
TG_ITOC	Integer to character grid time
TG_ITOF	Integers to file time
TG_RANG	Get range of times
TG_VALD	Compute valid time
TG_VTOF	Verification to forecast time

GRID TIME (TG) LIBRARY

Grid Time (TG) Library Summary

The GEMPAK GRID TIME library subroutines process grid times for GEMPAK.

The GEMPAK format for grid time is a character string:

YYMMDD/HHMMthhhmm

YY	last two digits of the year
MM	month
DD	day of month
/	date and time separator
HH	hour
MM	minute
t	type (F=forecast, A=analysis, G=guess, I=initialization)
hhh	forecast hour
mm	forecast minute

The string before the / is the DATE; the string after the / is the TIME; "thhhmm" is the FORECAST TIME.

If hhhmm is one or two characters, they will be interpreted as hours. Three or more characters will be right justified in hhhmm.

The forecast type may specify V for verification time. In this case, hhhmm will be subtracted from the DATE and TIME and the type will be returned as F.

A partial time may be entered for the grid time. The last time in the data set will be used to fill in the missing parts. If the input string does not contain the date and time separator, '/', the input string is assumed to be a left-justified time (i.e., 12 represents HH). If the forecast type is not specified, the forecast time from the last time will be used. FIRST and LAST can be used to specify the first and last times in the file.

If the last time in the file is 890408/1200F72, the abbreviated forms will be translated into the following GEMPAK times:

3/11	---->	890403/1100F72
00F00	---->	890408/0000F00
7/	---->	890407/1200F72

GRID TIME (TG) LIBRARY

LAST	---->	890408/1200F72
LASTF00	---->	890408/1200F00
F12	---->	890408/1200F12
9/00V48	---->	890407/1200F48

ERROR MESSAGES:

```
[TG  -1]  Invalid date or time.
[TG  -2]  Invalid forecast type.
[TG  -3]  Invalid forecast time.
[TG  -4]  No times in input file.
[TG  -5]  Invalid time range.
[TG  -6]  Single time invalid for range.
[TG  -7]  Invalid forecast type for ALL.
[TG  -8]  Invalid first time in range.
[TG  -9]  Invalid last time in range.
[TG -10]  First time is after last time.
[TG -11]  First and last times are the same.
[TG -12]  Forecast types must be the same for different dates.
[TG -13]  Forecast times must be the same for different dates.
[TG -14]  The time increment is invalid.
[TG -15]  Too many times in list.
[TG -16]  No times in range.
```

GRID TIME (TG) LIBRARY

TG Library Calls

```
TG_CFTM   ( ifcast, / ftype, ftime, iret )
TG_CTOI   ( gdattm, / intdtf, iret )
TG_DIFF   ( dattm1, dattm2, / nmin, iret )
TG_FLST   ( ntime, times, ntimf, filtim, / nfound, timfnd, iret )
TG_FTOI   ( iftime, / intdtf, iret )
TG_FULL   ( gdattm, firstm, lasttm, / fulltm, iret )
TG_IFTM   ( ftype, ftime, / ifcast, iret )
TG_ITOC   ( intdtf, / gdattm, iret )
TG_ITOF   ( intdtf, / iftime, iret )
TG_RANG   ( gdattm, ntimf, filtim, / ntime, times, iret )
TG_VALD   ( gdattm, / vdattm, iret )
TG_VTOF   ( intvdt, / intfdt, iret )
```

28.1 TG_CFTM - CREATE CHARACTER FORECAST TIME

This subroutine converts an integer grid forecast time into the character forecast type and time. The forecast type is A (analysis), F (forecast), G (guess) or I (initialize). If the forecast time is less than 100 and the minutes are 00, only hh is returned.

TG_CFTM (IFCAST, FTYPE, FTIME, IRET)

Input parameters:

IFCAST	INTEGER	GEMPAK grid time

Output parameters:

FTYPE	CHAR*1	Forecast type (A,F,G,I)
FTIME	CHAR*	Forecast time (hhhmm)
IRET	INTEGER	Return code 0 = normal return -2 = invalid forecast type -3 = invalid forecast time

28.2 TG_CTOI - GRID TIME TO INTEGERS

This subroutine converts a full grid date/time string into the three integers for date, time and forecast time.

TG_CTOI (GDATTM, INTDTF, IRET)

Input parameters:

GDATTM	CHAR*	GEMPAK grid date/time

Output parameters:

INTDTF (3)	INTEGER	Grid date, time, forecast
IRET	INTEGER	Return code 0 = normal return -2 = invalid forecast type -3 = invalid forecast time

28.3 TG_DIFF - GRID TIME DIFFERENCE

This subroutine computes the time difference in minutes between two GEMPAK grid times. The time difference is time1 - time2 and may be computed for a maximum of one year.

TG_DIFF (DATTM1, DATTM2, NMIN, IRET)

Input parameters:

DATTM1	CHAR*	First GEMPAK grid time
DATTM2	CHAR*	Second GEMPAK grid time

Output parameters:

NMIN	INTEGER	Difference in minutes
IRET	INTEGER	Return code 0 = normal return -12 = invalid time range

28.4 TG_FLST - FIND TIMES IN FILE

This subroutine takes a list of input times and determines which times are in a list of times in the file.

TG_FLST (NTIME, TIMES, NTIMF, FILTIM, NFOUND, TIMFND, IRET)

Input parameters:

NTIME	INTEGER	Number of times in input list
TIMES (NTIME)	CHAR*	Input times
NTIMF	INTEGER	Number of times in file
FILTIM (NTIMF)	CHAR*	Times in file

Output parameters:

NFOUND	INTEGER	Number of times found
TIMFND (NFOUND)	CHAR*	Times found
IRET	INTEGER	Return code 0 = normal return

28.5 TG_FTOI - FILE TIME TO INTEGERS

This subroutine converts the two integers stored in a grid file into three integers containing the date, time and forecast time.

TG_FTOI (IFTIME, INTDTF, IRET)

Input parameters:

IFTIME (2)	INTEGER	Grid time stored in file

Output parameters:

INTDTF (3)	INTEGER	Date, time, forecast time
IRET	INTEGER	Return code 0 = normal return

28.6 TG_FULL - PARTIAL TO FULL GRID TIME

This subroutine converts the user input for a single grid time into a full grid time string.

TG_FULL (GDATTM, FIRSTM, LASTTM, FULLTM, IRET)

Input parameters:

GDATTM	CHAR*	Input grid time
FIRSTM	CHAR*	First time in grid file
LASTTM	CHAR*	Last time in grid file

Output parameters:

FULLTM	CHAR*	Full GEMPAK grid time
IRET	INTEGER	Return code 0 = normal return -1 = invalid date or time -2 = invalid forecast type -3 = invalid forecast time

28.7 TG_IFTM - CREATE INTEGER FORECAST TIME

This subroutine converts the grid forecast type and time into an integer forecast time.

TG_IFTM (IFCAST, FTYPE, FTIME, IRET)

Input parameters:

FTYPE	CHAR*1	Forecast type (A,F,G)
FTIME	CHAR*	Forecast time (hhhmm)

Output parameters:

IFCAST	INTEGER	GEMPAK forecast time
IRET	INTEGER	Return code 0 = normal return -2 = invalid forecast type -3 = invalid forecast time

28.8 TG_ITOC - INTEGER TO CHARACTER GRID TIME

This subroutine converts an integer time array containing the date, time and forecast time into a GEMPAK grid time.

TG_ITOC (INTDTF, GDATTM, IRET)

Input parameters:

INTDTF (3)	INTEGER	Date, time, forecast time

Output parameters:

GDATTM	CHAR*	GEMPAK grid time
IRET	INTEGER	Return code 0 = normal return -1 = invalid date or time

28.9 TG_ITOF - INTEGERS TO FILE TIME

This subroutine converts three integers containing the date, time and forecast field into the two integers stored in a grid file.

TG_ITOF (INTDTF, IFTIME, IRET)

Input parameters:		
INTDTF (3)	INTEGER	Date, time, forecast time
Output parameters:		
IFTIME (2)	INTEGER	Grid time stored in file
IRET	INTEGER	Return code 0 = normal return

28.10 TG_RANG - GET RANGE OF TIMES

This subroutine converts the user input for a grid time range into a list of times.

TG_RANG (GDATTM, NTIMF, FILTIM, NTIME, TIMES, IRET)

Input parameters:

GDATTM	CHAR*	Input grid time
NTIMF	INTEGER	Number of times in file
FILTIM (NTIMF)	CHAR*	Times in file

Output parameters:

NTIME	INTEGER	Number of times selected
TIMES (NTIME)	CHAR*	Selected times
IRET	INTEGER	Return code 0 = normal return -4 = no times in file -5 = invalid time range -6 = single time invalid -7 = invalid forecast type -8 = invalid first time -9 = invalid last time -10 = first time after last -11 = first and last are same -12 = forecast types not same -13 = forecast times not same -14 = invalid time increment -15 = too many times in list -16 = no times in range

28.11 TG_VALD - COMPUTE VALID TIME

This subroutine converts a grid time to the valid date/time. The input string must be a full grid time. The output time will contain only the date and time. The input and output strings may be the same.

TI_FTOV (GDATTM, VDATTM, IRET)

Input parameters:		
GDATTM	CHAR*	Grid date/time
Output parameters:		
VDATTM	CHAR*	Valid date/time
IRET	INTEGER	Return code 0 = normal return -1 = invalid time -2 = invalid forecast type -3 = invalid forecast time

28.12 TG_VTOF - VERIFICATION TO FORECAST TIME

This subroutine converts an integer grid time in V syntax to an integer time in F syntax.

TG_VTOF (INTVDT, INTFDT, IRET)

Input parameters:

INTVDT (3)	INTEGER	Date, time, fcast as V

Output parameters:

INTFDT (3)	INTEGER	Date, time, fcast as F
IRET	INTEGER	Return code 0 = normal return

CHAPTER 29

TIME (TI) LIBRARY

TI_ADDD	Add one day
TI_CDTM	GEMPAK time to date and time
TI_CTOI	GEMPAK time to integer time
TI_DAYW	Day of week
TI_DIFD	Time difference in days
TI_DIFF	Time difference in minutes
TI_FIND	Find user input times
TI_GREN	Local time to UTC
TI_GRTM	Integer local time to UTC
TI_GTIM	System local time
TI_IDTM	Date and time to GEMPAK time
TI_ITOC	Integer array to GEMPAK time
TI_MDIF	Time difference from integer time
TI_SORT	Sort times
TI_STNT	Compute station time
TI_SUBD	Subtract one day

Time (TI) Library Summary

The GEMPAK TIME library subroutines process times for GEMPAK.

The GEMPAK format for time is a character string, YYMMDD/HHMM where:

YY is the last two digits of the year
MM is the month
DD is the day of the month
/ is the date and time separator
HH is the hour
MM is the minutes past the hour

The string before the / is the DATE; the string after the / is the TIME.

A partial time may be entered in TI_FIND. The latest date/time in the data set will be used to fill in the missing parts. If the input string does not contain the date and time separator, '/', the input string is assumed to be a left-justified time (i.e., 12 is 1200 UTC). For example:

If the last time in the file is '840515/1200 ', the abbreviated forms will be translated into the following GEMPAK times:

13/11	---->	840513/1100
13	---->	840515/1300
13/	---->	840513/1200
0412/1300	---->	840412/1300

In addition to the above date/time format, there are four symbols which may be entered in TI_FIND in place of a specific date/time:

LAST returns the most recent time available
LIST displays a list of the available times
ALL returns all the available times
/ALL returns all the times for a particular day

In some GEMPAK subroutines, especially those doing real-time data ingest, it is more convenient to store the date/time information as integers. In those subroutines, 5 integers are used to store the time:

```
idtarr (1) = YYYY
idtarr (2) = MM
idtarr (3) = DD
idtarr (4) = HH
idtarr (5) = MM
```

The subroutines TI_CTOI and TI_ITOC are provided to translate between integer and character times.

Similarly, GEMPAK times are often stored in files as two integers representing the date and time. Subroutines TI_CDTM and TI_IDTM translate between integer and character times.

Note that grid times are now processed in the TG library.

ERROR MESSAGES:

```
[TI   2]  The time ... is not found in the dataset.
[TI   1]  EXIT entered by the user.
[TI  -1]  The DATTIM ... is invalid.
[TI  -2]  No valid times entered.
[TI  -3]  No times within range.
[TI  -4]  "LIST" cannot be entered when RESPOND is false.
[TI  -5]  There are no times in the dataset.
[TI  -7]  The value for the year is invalid.
[TI  -8]  The value for the month is invalid.
[TI  -9]  The value for the day is invalid.
[TI -10]  The value for the hour is invalid.
[TI -11]  The value for the minute is invalid.
[TI -12]  Invalid time difference.
[TI -13]  Invalid year for conversion to Greenwich time.
[TI -14]  Invalid forecast time.
```

TIME (TI) LIBRARY

TI Library Calls

```
TI_ADDD  ( idtarr, / jdtarr, iret )
TI_CDTM  ( idate, itime, / dattim, iret )
TI_CTOI  ( dattim, / idtarr, iret )
TI_DAYW  ( idtarr, / idayw, iret )
TI_DIFD  ( dattm1, dattm2, / days, iret )
TI_DIFF  ( dattm1, dattm2, / nmin, iret )
TI_FIND  ( dattim, ntimin, timlst, / timout, ntime, timfnd, iret )
TI_GREN  ( locarr, / igrarr, iret )
TI_GRTM  ( locarr, ioflst, iofldt, / igrarr, iret )
TI_GTIM  ( / dattim, iret )
TI_IDTM  ( dattim, / idate, itime, iret )
TI_ITOC  ( idtarr, / dattim, iret )
TI_MDIF  ( idtar1, idtar2, / nmin, iret )
TI_SORT  ( ntime, timin, / outime, iret )
TI_STNT  ( dattim, ihhmm, / stntim, iret )
TI_SUBD  ( idtarr, / jdtarr, iret )
```

29.1 TI_ADDD - ADD ONE DAY

This subroutine adds a day to the time in an integer array. The input and output arrays may be the same array.

TI_ADDD (IDTARR, JDTARR, IRET)

Input parameters:

IDTARR (5)	INTEGER	Time array (YYYY,MM,DD,HH,MM)

Output parameters:

JDTARR (5)	INTEGER	Time array (YYYY,MM,DD,HH,MM)
IRET	INTEGER	Return code 0 = normal return -7 = invalid year -8 = invalid month -9 = invalid day

29.2 TI_CDTM - GEMPAK TIME TO DATE AND TIME

This subroutine converts an integer date (YYMMDD) and time (HHMM) into a standard GEMPAK time.

TI_CDTM (IDATE, ITIME, DATTIM, IRET)

Input parameters:

IDATE	INTEGER	Date (YYMMDD)
ITIME	INTEGER	Time (HHMM)

Output parameters:

DATTIM	CHAR*	GEMPAK time
IRET	INTEGER	Return code 0 = normal return -1 = invalid time -7 = invalid year -8 = invalid month -9 = invalid day -10 = invalid hour -11 = invalid minute

29.3 TI_CTOI - GEMPAK TIME TO INTEGER TIME

This subroutine converts a standard GEMPAK time into an integer array containing year, month, day, hour, and minute. The input string must be a complete GEMPAK date/time. The integers are checked for validity. If YY is less than 20, 2000 is added for the year; otherwise, 1900 is added.

TI_CTOI (DATTIM, IDTARR, IRET)

Input parameters:

DATTIM	CHAR*	GEMPAK time

Output parameters:

IDTARR (5)	INTEGER	Time array (YYYY,MM,DD,HH,MM)
IRET	INTEGER	Return code 0 = normal return -1 = invalid time -7 = invalid year -8 = invalid month -9 = invalid day -10 = invalid hour -11 = invalid minute

29.4 TI_DAYW - DAY OF WEEK

This subroutine returns the day of week, IDAYW, given an integer time. IDAYW is set to 1 for Sunday, ..., 7 for Saturday. The algorithm is only valid after 1600 and may need to be modified in 2000. This subroutine does not check that a valid time was entered. If the year is less than 20, 2000 is added. If the year is greater than 20, but less than 100, 1900 is added.

TI_DAYW (IDTARR, IDAYW, IRET)

Input parameters:

IDTARR (5)	INTEGER	Time array (YYYY,MM,DD,HH,MM)

Output parameters:

IDAYW	INTEGER	Day of week (1 = Sun,...)
IRET	INTEGER	Return code 0 = normal return

29.5 TI_DIFD - TIME DIFFERENCE IN DAYS

This subroutine computes the time difference in days between two standard GEMPAK times. The time difference is time1 - time2.

TI_DIFD (DATTM1, DATTM2, DAYS, IRET)

Input parameters:

DATTM1	CHAR*	First GEMPAK time
DATTM2	CHAR*	Second GEMPAK time

Output parameters:

DAYS	REAL	Difference in days
IRET	INTEGER	Return code 0 = normal return -12 = invalid time range

29.6 TI_DIFF - TIME DIFFERENCE IN MINUTES

This subroutine computes the time difference in minutes between two standard GEMPAK times. The time difference is time1 - time2 and may be computed for a maximum of one year.

TI_DIFF (DATTM1, DATTM2, NMIN, IRET)

Input parameters:

DATTM1	CHAR*	First GEMPAK time
DATTM2	CHAR*	Second GEMPAK time

Output parameters:

NMIN	INTEGER	Difference in minutes
IRET	INTEGER	Return code 0 = normal return -12 = invalid time range

29.7 TI_FIND - FIND USER INPUT TIMES

This subroutine converts the user input for DATTIM into a list of times. The times may be entered as a list or range of times. The requested times are returned in TIMFND. The times in the file are input in TIMLST, where the times must be in the standard GEMPAK format and must be sorted from earliest to latest. This subroutine will write error messages for any error encountered.

```
TI_FIND ( DATTIM, NTIMIN, TIMLST, TIMOUT, NTIME, TIMFND,
          IRET )
```

Input parameters:

DATTIM	CHAR*	User input time
NTIMIN	INTEGER	Number of data set times
TIMLST (NTIMIN)	CHAR*	Data set times

Output parameters:

TIMOUT	CHAR*	User input time
NTIME	INTEGER	Number of times requested
TIMFND (NTIME)	CHAR*	Requested times
IRET	INTEGER	Return code +2 = time not in data set +1 = EXIT entered 0 = normal return -1 = DATTIM is invalid -2 = no valid times entered -3 = no times in range -4 = cannot list times -5 = data set has no times

29.8 TI_GREN - LOCAL TIME TO UTC

This subroutine converts an integer local time to UTC (Universal Time Coordinated), formerly GMT (Greenwich Mean Time). The offsets to UTC (GMT) must be stored in GEMPRM.PRM . The subroutine is valid through 1999. Daylight time is computed using the rule valid in 1987. The input and output time array names may be the same.

TI_GREN (LOCARR, IGRARR, IRET)

Input parameters:

LOCARR (5)	INTEGER	Local time (YYYY,MM,DD,HH,MM)

Output parameters:

IGRARR (5)	INTEGER	UTC time (YYYY,MM,DD,HH,MM)
IRET	INTEGER	Return code 0 = normal return -13 = invalid year

29.9 TI_GRTM - INTEGER LOCAL TIME TO UTC

This subroutine converts an integer local time to UTC (Universal Time Coordinated), formerly GMT (Greenwich Mean Time). The subroutine is valid through 1999. Daylight time is computed using the rule valid in 1987. The input and output time array names may be the same.

TI_GRTM (LOCARR, IOFLST, IOFLDT, IGRARR, IRET)

Input parameters:

LOCARR (5)	INTEGER	Local time (YYYY,MM,DD,HH,MM)
IOFLST	INTEGER	Standard time offset
IOFLDT	INTEGER	Daylight time offset

Output parameters:

IGRARR (5)	INTEGER	UTC time (YYYY,MM,DD,HH,MM)
IRET	INTEGER	Return code 0 = normal return -13 = invalid year

29.10 TI_GTIM - SYSTEM LOCAL TIME

This subroutine returns the current system clock time as a GEMPAK time.

TI_GTIM (DATTIM, IRET)

Output parameters:

DATTIM	CHAR*	System time in GEMPAK format
IRET	INTEGER	Return code 0 = normal return -1 = invalid time

29.11 TI_IDTM - DATE AND TIME TO GEMPAK TIME

This subroutine converts a standard GEMPAK time into an integer date (YYMMDD) and time (HHMM).

TI_IDTM (DATTIM, IDATE, ITIME, IRET)

Input parameters:

DATTIM	CHAR*	GEMPAK time

Output parameters:

IDATE	INTEGER	Date (YYMMDD)
ITIME	INTEGER	Time (HHMM)
IRET	INTEGER	Return code 0 = normal return -1 = invalid time -7 = invalid year -8 = invalid month -9 = invalid day -10 = invalid hour -11 = invalid minute

29.12 TI_ITOC - INTEGER ARRAY TO GEMPAK TIME

This subroutine converts an integer time array into a standard GEMPAK time. The integers are checked for validity.

TI_ITOC (IDTARR, DATTIM, IRET)

Input parameters:

IDTARR (5)	INTEGER	Time array (YYYY,MM,DD,HH,MM)

Output parameters:

DATTIM	CHAR*	GEMPAK time
IRET	INTEGER	Return code 0 = normal return -7 = invalid year -8 = invalid month -9 = invalid day -10 = invalid hour -11 = invalid minute

29.13 TI_MDIF - TIME DIFFERENCE FROM INTEGER TIME

This subroutine computes the time difference in minutes between two integer times. The time difference is time1 - time2 and may be computed for a maximum of one year.

TI_MDIF (IDTAR1, IDTAR2, NMIN, IRET)

Input parameters:

IDTAR1 (5)	INTEGER	Time array 1 (YYYY,MM,DD,HH,MM)
IDTAR2 (5)	INTEGER	Time array 2 (YYYY,MM,DD,HH,MM)

Output parameters:

NMIN	INTEGER	Difference in minutes
IRET	INTEGER	Return code 0 = normal return -12 = invalid time range

29.14 TI_SORT - SORT TIMES

This subroutine sorts a list of times from earliest to latest. The input and output arrays may be the same.

TI_SORT (NTIME, TIMIN, OUTIME, IRET)

Input parameters:

NTIME	INTEGER	Number of times
TIMIN (NTIME)	CHAR*	GEMPAK times

Output parameters:

OUTIME (NTIME)	CHAR*	Sorted times
IRET	INTEGER	Return code 0 = normal return

29.15 TI_STNT - COMPUTE STATION TIME

This subroutine returns the station observation time given the standard GEMPAK nominal time and the hour and minute of the observation.

TI_STNT (DATTIM, IHHMM, STNTIM, IRET)

Input parameters:

DATTIM	CHAR*	Nominal time
IHHMM	INTEGER	Observation hour/minute

Output parameters:

STNTIM	CHAR*	Station time
IRET	INTEGER	Return code 0 = normal return -7 = invalid year -8 = invalid month -9 = invalid day -10 = invalid hour -11 = invalid minute

29.16 TI_SUBD - SUBTRACT ONE DAY

This subroutine subtracts a day from the time in an integer array. The input and output arrays may be the same array.

TI_SUBD (IDTARR, JDTARR, IRET)

Input parameters:

IDTARR (5)	INTEGER	Time array (YYYY,MM,DD,HH,MM)

Output parameters:

JDTARR (5)	INTEGER	Time array (YYYY,MM,DD,HH,MM)
IRET	INTEGER	Return code 0 = normal return -7 = invalid year -8 = invalid month -9 = invalid day

CHAPTER 30

TERMINAL (TM) LIBRARY

TM_ACCP	Wait for user to accept values
TM_CHAR	Read strings from the terminal
TM_INT	Read integers from the terminal
TM_PAGE	Clear the terminal screen
TM_PROM	Write message to the terminal
TM_RCHR	Read user terminal input
TM_REAL	Read reals from the terminal
TM_STR	Read a string from the terminal
TM_WAIT	Wait for specified time interval
TM_WCR	Write message and wait for a <CR>

Terminal (TM) Library Summary

The TERMINAL library provides subroutines to write messages to the user's terminal and to read terminal input.

The subroutines TM_INT, TM_REAL and TM_CHR write a message to the terminal and return the integer, real or character array that was input by the user. TM_STR returns a single, unprocessed string. TM_ACCP waits for the user to respond by entering a carriage return. In every case, the user may type EXIT to stop processing.

Subroutines to clear the terminal screen, halt program execution and write messages are also available.

ERROR MESSAGES:

[TM 3] Too many values entered.
[TM 2] EXIT typed by user.
[TM 1] Carriage return entered by user.
[TM -3] Invalid input string.

TERMINAL (TM) LIBRARY

TM Library Calls

```
TM_ACCP    ( / iret )
TM_CHAR    ( messg, pagflg, newln, nexp, / chrstr, nstr, iret )
TM_INT     ( messg, pagflg, newln, nexp, / integ, nint, iret )
TM_PAGE    ( / iret )
TM_PROM    ( messg, pagflg, newlin, / iret )
TM_RCHR    ( / string, iret )
TM_REAL    ( messg, pagflg, newln, nexp, / rlnos, nreal, iret )
TM_STR     ( messg, pagflg, newln, / string, iret )
TM_WAIT    ( isecnd, / iret )
TM_WCR     ( / iret )
```

30.1 TM_ACCP - WAIT FOR USER TO ACCEPT VALUES

This subroutine writes the following message at user's terminal:

'Enter <CR> to accept parameters or type EXIT:'

The user must enter either <cr> or EXIT.

TM_ACCP (IRET)

Output parameters:

IRET	INTEGER	Return code 2 = EXIT entered 1 = <cr> entered 0 = normal return

30.2 TM_CHAR - READ STRINGS FROM THE TERMINAL

This subroutine writes a message to the user's terminal followed by 'or type EXIT'. The phrase '<CR> to page' may also be added. An array of character strings entered by the user is returned.

TM_CHAR (MESSG, PAGFLG, NEWLN, NEXP, CHRSTR, NSTR, IRET)

Input parameters:

MESSG	CHAR*	Message
PAGFLG	LOGICAL	Flag to add '<CR> to page'
NEWLN	LOGICAL	Flag to move to new line
NEXP	INTEGER	Maximum # of strings to return

Output parameters:

CHRSTR (NSTR)	CHAR*	User input strings
NSTR	INTEGER	Number of strings returned
IRET	INTEGER	Return code 3 = too many strings 2 = EXIT entered 1 = <CR> entered 0 = normal return

30.3 TM_INT - READ INTEGERS FROM THE TERMINAL

This subroutine writes a message to the user's terminal followed by 'or type EXIT'. The phrase '<CR> to page' may also be added. An array of integers entered by the user is returned.

TM_INT (MESSG, PAGFLG, NEWLN, NEXP, INTEG, NINT, IRET)

Input parameters:

MESSG	CHAR*	Message
PAGFLG	LOGICAL	Flag to add '<CR> to page'
NEWLN	LOGICAL	Flag to move to new line
NEXP	INTEGER	Maximum number of integers

Output parameters:

INTEG (NINT)	INTEGER	Integers entered by user
NINT	INTEGER	Number of integers
IRET	INTEGER	Return code 3 = too many integers 2 = EXIT entered 1 = <CR> entered 0 = normal return -3 = invalid input string

30.4 TM_PAGE - CLEAR THE TERMINAL SCREEN

This subroutine clears the terminal screen.

TM_PAGE (IRET)

Output parameters:

IRET	INTEGER	Return code 0 = normal return

30.5 TM_PROM - WRITE MESSAGE TO THE TERMINAL

This subroutine writes a message to the user's terminal followed by the phrase 'or type EXIT'. The phrase '<CR> to page' may also be added. This subroutine does not wait for a user response.

TM_PROM (MESSG, PAGFLG, NEWLIN, IRET)

Input parameters:

MESSG	CHAR*	Message
PAGFLG	LOGICAL	Flag to add '<CR> to page'
NEWLIN	LOGICAL	Flag to move to new line

Output parameters:

IRET	INTEGER	Return code 0 = normal return

30.6 TM_RCHR - READ USER TERMINAL INPUT

This subroutine reads a character string from the terminal and checks for <CR> or EXIT.

TM_RCHR (STRING, IRET)

Output parameters:

STRING	CHAR*	User input
IRET	INTEGER	Return code 2 = EXIT entered 1 = <CR> entered 0 = normal return

30.7 TM_REAL - READ REALS FROM THE TERMINAL

This subroutine writes a message to the user's terminal followed by 'or type EXIT'. The phrase '<CR> to page' may also be added. An array of real numbers entered by the user is returned.

TM_REAL (MESSG, PAGFLG, NEWLN, NEXP, RLNOS, NREAL, IRET)

Input parameters:

MESSG	CHAR*	Message
PAGFLG	LOGICAL	Flag to add '<CR> to page'
NEWLN	LOGICAL	Flag to move to new line
NEXP	INTEGER	Maximum number of real numbers

Output parameters:

RLNOS (NREAL)	REAL	Real numbers entered
NREAL	INTEGER	Number of real numbers
IRET	INTEGER	Return code 3 = too many reals 2 = EXIT entered 1 = <CR> entered 0 = normal return -3 = invalid input string

30.8 TM_STR - READ A STRING FROM THE TERMINAL

This subroutine writes a message to the user's terminal followed by 'or type EXIT'. The phrase '<CR> to page' may also be added. The string entered by the user is returned.

TM_STR (MESSG, PAGFLG, NEWLN, STRING, IRET)

Input parameters:

MESSG	CHAR*	Message
PAGFLG	LOGICAL	Flag to add '<CR> to page'
NEWLN	LOGICAL	Flag to move to new line

Output parameters:

STRING	CHAR*	String entered
IRET	INTEGER	Return code 2 = EXIT entered 1 = <CR> entered 0 = normal return

30.9 TM_WAIT - WAIT FOR SPECIFIED TIME INTERVAL

This subroutine halts the execution of the calling program for up to 420 seconds (7 minutes).

TM_WAIT (ISECND, IRET)

Input parameters:

ISECND	INTEGER	Length of time in seconds

Output parameters:

IRET	INTEGER	Return code 0 = normal return

30.10 TM_WCR - WRITE MESSAGE AND WAIT FOR A <CR>

This subroutine prompts the user with the message,

'Enter <CR> to continue'

and waits for the user to enter a carriage return.

TM_WCR (IRET)

Output parameters:

IRET	INTEGER	Return code 0 = normal return

APPENDIX A

GEMPAK CONSTANTS

This appendix contains the parameter definitions for the GEMPAK software which are contained in GEMINC:GPARMS.PRM .

```
C***********************************************************************
C* GEMPRM.PRM
C*
C* This include file contains parameter definitions for the GEMPAK
C* software.
C*
C***********************************************************************
C!
C! Missing data definitions
C!
      PARAMETER      ( RMISSD = -9999.0 )        Missing data value
C!
      PARAMETER      ( RDIFFD =  0.1    )        Missing integer value
C!
      PARAMETER      ( IMISSD = -9999   )        Missing value fuzziness
C!
      LOGICAL          ERMISS                    Declare for stmt func
C!
C!
C! Physical and mathematical constants
C!
      PARAMETER      ( PI = 3.14159265  )
      PARAMETER      ( HALFPI = PI / 2. )
      PARAMETER      ( TWOPI  = 2. * PI )
      PARAMETER      ( PI4TH  = PI / 4. )        PI,...
C!
      PARAMETER      ( DTR = PI / 180.  )
      PARAMETER      ( RTD = 180. / PI  )        Degrees <--> Radians
C!
      PARAMETER      ( RADIUS = 6371200. )
```

GEMPAK CONSTANTS

```
C!                                                          Earth radius
        PARAMETER       ( OMEGA   = 7.292E-5 )
C!                                                          Earth angular veclocity
        PARAMETER       ( GRAVTY  = 9.80616  )
C!                                                          Acceleration of gravity
        PARAMETER       ( RDGAS   = 287.04   )
        PARAMETER       ( RKAP    = RDGAS / GRAVTY )
C!                                                          Gas constant of dry air
        PARAMETER       ( RKAPPA  = 2. / 7. )
        PARAMETER       ( AKAPPA  = 7. / 2. )
C!                                                          Poisson constant;inverse
        PARAMETER       ( GAMUSD  = 6.5 )
C!                                                          US std atmos lapse rate
C!
C! File information parameters
C!
        PARAMETER       ( MMKEY   =     12    )
C!                                                          Maximum # of keys
        PARAMETER       ( MMHDRS  =   4000    )
C!                                                          Maximum # of headers
        PARAMETER       ( MMPRT   =     20    )
C!                                                          Maximum # of parts
        PARAMETER       ( MMLIST  =     20    )
C!                                                          Maximum search list
        PARAMETER       ( MMFREE  =     62    )
C!                                                          Number of free pairs
        PARAMETER       ( MMFILE  =      3    )
C!                                                          Maximum # of open files
        PARAMETER       ( MBLKSZ  =    128    )
C!                                                          Block size
        PARAMETER       ( MCACHE  =      8    )
C!                                                          # of cached records
        PARAMETER       ( MMPARM  =     40    )
C!                                                          Maximum # of parameters
        PARAMETER       ( MMFHDR  =     10    )
C!                                                          Maximum # of file hdrs
        PARAMETER       ( MMSRCH  =     30    )
C!                                                          Max # of cond searches
        PARAMETER       ( MTVAX   =      2    )
        PARAMETER       ( MTSUN   =      3    )
        PARAMETER       ( MTIRIS  =      4    )
        PARAMETER       ( MTMACH  = MTVAX     )
C!                                                          Machine type
        PARAMETER       ( MMFLDP  = MMFILE * MMPRT )
C!
        PARAMETER       ( MDREAL  =      1    )
        PARAMETER       ( MDINTG  =      2    )
        PARAMETER       ( MDCHAR  =      3    )
        PARAMETER       ( MDRPCK  =      4    )
        PARAMETER       ( MDGRID  =      5    )
C!                                                          Data types in DM lib
```

GEMPAK CONSTANTS

```
        PARAMETER       ( MDGNON =      0   )
        PARAMETER       ( MDGGRB =      1   )
        PARAMETER       ( MDGNMC =      2   )
        PARAMETER       ( MDGDIF =      3   )
        PARAMETER       ( MDGDEC =      4   )
C!                                              Grid packing types
        PARAMETER       ( MFSF =        1   )
        PARAMETER       ( MFSN =        2   )
        PARAMETER       ( MFGD =        3   )
C!                                              Data file types
        PARAMETER       ( MFNONE =      0   )
        PARAMETER       ( MFAIRW   =     1  )
        PARAMETER       ( MFMETR =      2   )
        PARAMETER       ( MFSHIP =      3   )
C!                                              Unknown, airways, metar,
C!                                              ship data source
        PARAMETER       ( MFBUDY =      4   )
        PARAMETER       ( MFSYNP =      5   )
        PARAMETER       ( MFRAOB =      4   )
        PARAMETER       ( MFVAS    =    5   )
C!                                              Raob, VAS data source
        PARAMETER       ( MMRECL =      1   )
C!                                              Multiplier for RECL in
C!                                              file create/open
C!                                              (usually 4 on UNIX sys)
C!
C! Declarations for array sizes in programs
C!
        PARAMETER       ( LLMXLV =    500    )
C!                                              Max # levels/station
        PARAMETER       ( LLMXTM =    200    )
C!                                              Max # times/dataset
        PARAMETER       ( LLMXGT =   1000    )
C!                                              Max # grid times
        PARAMETER       ( LLMXST =     20    )
C!                                              Max # stations in list
        PARAMETER       ( LLMXDT = MMPARM * LLMXLV )
C!                                              Max # data points
        PARAMETER       ( LLSTFL =   2000    )
C!                                              Max # stations in file
        PARAMETER       ( LLMXGD =   8000    )
C!                                              Max # grid points
        PARAMETER       ( LLSTHL =     20    )
C!                                              Max header size
        PARAMETER       ( LLGDHD =    128    )
C!                                              Max grid hdr length
        PARAMETER       ( LLOAGD =    400    )
C!                                              Max # grids from 1 OA
        PARAMETER       ( LLCLEV =     50    )
C!                                              Max # of contour lvls
        PARAMETER       ( LLAXIS =     32    )
```

```
C!                                                     Max # of axis labels
C!
C! Offsets from local to UTC (GMT) time in HHMM (hour/minute) format
C!
      PARAMETER        ( JOFLST =   500   )
C!                                                 Offset for UTC/EST
      PARAMETER        ( JOFLDT =   400   )
C!                                                 Offset for UTC/EDT
C!
C! GEMPAK table files
C!
      CHARACTER*(*)    GEOTBL, PCVTBL, PRMFLG, SFSTBL, SNSTBL
      CHARACTER*(*)    SNWTBL, SATNAV, GRDNAV
C!
      PARAMETER ( GEOTBL = 'GEMTABL:GEOG.TBL'    )
C!                                                     Geographic
      PARAMETER ( PCVTBL = 'GEMTABL:PCCONV.TBL' )
C!                                                     Parameter conv
      PARAMETER ( SNSTBL = 'GEMTABL:SNSTNS.TBL' )
C!                                                     Raob stations
      PARAMETER ( SNWTBL = 'GEMTABL:SNWORLD.TBL' )
C!                                                     World raob stns
      PARAMETER ( SFSTBL = 'GEMTABL:SFSTNS.TBL' )
C!                                                     Surface stations
      PARAMETER ( PRMFLG = 'GEMTABL:PRMFLG.TBL' )
C!                                                     Parameter flags
      PARAMETER ( SATNAV = 'GEMTABL:SATNAV.TBL' )
C!                                                     AOIPS satellite
      PARAMETER ( GRDNAV = 'GEMTABL:GRDNAV.TBL' )
C!                                                     Grid navigation
C!
C! File Path names
C!
      CHARACTER        GEMERR*7, GEMTBL*8, GEMEXE*7, GEMGLB*17
      CHARACTER        GEMHLP*7, MAPLOC*8, GPLERR*17
C!
      PARAMETER        ( GEMERR = 'GEMERR:' )
C!
      PARAMETER        ( GEMTBL = 'GEMTABL:' )
C!
      PARAMETER        ( GEMEXE = 'GEMEXE:' )
C!
      PARAMETER        ( GEMGLB = 'GEMEXE:GEMGLB.PDF')
C!
      PARAMETER        ( GEMHLP = 'GEMHLP:' )
C!
      PARAMETER        ( MAPLOC = 'GEMMAPS:')
C!
      PARAMETER        ( GPLERR = 'GEMERR:GEMPLT.ERR' )
C!
C! ASCII character constants
```

```
C!
        CHARACTER*1     CHNULL, CHSPAC, CHTAB, CHESC, CHFS, CHUS, CHGS
        CHARACTER*1     CHCR, CHLF, CHFF, CHCAN
C!
        PARAMETER       ( CHNULL = CHAR (0) )
C!                                              Null
        PARAMETER       ( CHTAB  = CHAR (9) )
C!                                              Tab
        PARAMETER       ( CHLF   = CHAR (10) )
C!                                              Line feed
        PARAMETER       ( CHFF   = CHAR (12) )
C!                                              Form feed
        PARAMETER       ( CHCAN  = CHAR (24) )
C!                                              Cancel (CAN)
        PARAMETER       ( CHCR   = CHAR (13) )
C!                                              Carriage return
        PARAMETER       ( CHESC  = CHAR (27) )
C!                                              Escape
        PARAMETER       ( CHFS   = CHAR (28) )
C!                                              FS
        PARAMETER       ( CHGS   = CHAR (29) )
C!                                              GS
        PARAMETER       ( CHUS   = CHAR (31) )
C!                                              US
        PARAMETER       ( CHSPAC = CHAR (32) )
C!                                              Space
C!
C---------------------- COORDINATE SYSTEMS ----------------------------
        CHARACTER       sysup*8, syslo*8
        PARAMETER       ( sysup = 'DNVPLMIG', syslo = 'dnvplmig' )
        CHARACTER       carray (8)*1
        COMMON          / GPSYS / carray
C
C-----------------------GPLT BUFFER SIZE--------------------------------
        PARAMETER       ( IGBSIZ  = 100000 )
        PARAMETER       ( IGTBSZ  = 1100   )
C
C-----------------------CYLINDRICAL MAP TRANSFORMATIONS-----------------
        PARAMETER       ( MCCYL = 1 )
        PARAMETER       ( MPCEQU = 1, MPCMER = 2, MPCMCD = 3 )
        PARAMETER       ( MSCEQU = 1 )
C
C-----------------------AZIMUTHAL MAP TRANSFORMATIONS-------------------
        PARAMETER       ( MCAZM = 2 )
        PARAMETER       ( MPAEQU = 1, MPASTR = 2, MPAORT = 3,
     +                            MPALAM = 4 )
        PARAMETER       ( MPAGNO = 5 )
        PARAMETER       ( MSANOR = 1, MSASOU = 2 )
C
C-----------------------CONICAL MAP TRANSFORMATIONS---------------------
        PARAMETER       ( MCCON = 3 )
```

```
      PARAMETER          ( MPCNOR = 1, MPCSOU = 2 )
C
C-----------------------OBLIQUE MERCATOR-------------------------------
        PARAMETER         ( MCMER = 4 )
        PARAMETER         ( MPTMER = 1, MPUTM = 2, MPOBLQ = 3 )
C
C-----------------------SATELLITE TRANSFORMATIONS----------------------
      PARAMETER          ( MCGOES = 6 , MPVAS = 1, MPAOI = 2,
     +                            MPNPGS = 3 )
```

APPENDIX B

CHANGES FROM GEMPAK4 TO GEMPAK5

This appendix describes the changes in the GEMPAK5 libraries from GEMPAK4.

DG library: The following subroutines have been added:
DG_OFIL - opens multiple files
DG_AREA - defines a subset area
DG_FLNO - returns grid file number
DG_OANG - sets orientation angle for cross section
DG_VECR - returns grid relative vector

RESPND: The input parameter RESPND has been removed from the following subroutines:
DG_GRID, DG_VECT, GR_LIST, TI_FIND

Grid packing: The following subroutines have been added to pack and unpack grids:
DP_PDEC, DP_PDIF, DP_PGRB, DP_UDIF, DP_UGRB, DP_UNMC, GD_WPGD, GD_WPPG

GR library: The following subroutines are new:
GR_AXLV, GR_INTP, GR_PACK, GR_PLIN, GR_PLOC, GR_ROBS
GR_GTIM replaced GR_TIME

IN library: The following subroutines are new:
IN_AXIS, IN_CINT, IN_LINE, IN_MRGD, IN_PRMC, IN_SKYC, IN_TAXS, IN_WSYM

OA library: The objective analysis has a first guess capability added:
OA_GUES, OA_NAVG

PD library: These subroutines have been added to allow the grid diagnostics to compute meteorological parameters efficiently.

PR library: The following subroutines have been added:
PR_AMSL replaces PR_SALT, PR_COMT, PR_D100, PR_HGFS,

CHANGES FROM GEMPAK4 TO GEMPAK5

PR_HGSF, PR_INMM, PR_M100, PR_MMIN

Station report time: This time has been changed from a character GEMPAK time to an integer, IHHMM, representing the hour and minute of the report. The following subroutines are affected: SF_RDAT, SN_RDAT, PC_SSTN, PC_STIM

SS library: The following routines have been replaced by system independent FL modules: SS_FLUN, SS_GLUN
SS_EXIT has been added to terminate a program. It replaces the call to EXIT.

TB library: TB_GRNV is a new subroutine to read a grid navigation table.
TB_GCXS has been deleted.

Grid time library: All the subroutines which deal with grid times have been removed from the TI library and moved to the new TG library.

Miscellaneous new subroutines:

FL_INQR
GD_NGRD replaced GD_NMGD
LC_FLOC
LV_CCRD
PT_WSYM
SF_UARE
ST_RMST
TI_DIFD

Miscellaneous deleted subroutine:

TM_WMSG

☆U.S. GOVERNMENT PRINTING OFFICE: 1991-527-064/26006

NASA National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-4261, Part 1		

4. Title and Subtitle	5. Report Date
GEMPAK5 Part 1--GEMPAK5 Programmer's Guide Version 5.0	February 1991
	6. Performing Organization Code 973
7. Author(s) Mary L. desJardins, Keith F. Brill, and Steven S. Schotz	8. Performing Organization Report No. 91B00036
	10. Work Unit No.
9. Performing Organization Name and Address Laboratory for Hydrospheric Processes NASA/Goddard Space Flight Center Greenbelt, Maryland 20771	11. Contract or Grant No.
	13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546-0001	14. Sponsoring Agency Code

15. Supplementary Notes

Mary L. desJardins: NASA/GSFC, Greenbelt, MD, 20771
Keith F. Brill: NOAA/NWS/NMC, Washington, DC.
Steven S. Schotz: General Sciences Corporation, Lanham, MD, 20785.

16. Abstract

GEMPAK is a general meteorological software package used to analyze and display conventional meteorological data as well as satellite derived parameters. This Programmer's Guide described the subroutines which can be used to build new GEMPAK programs. Part 1 contains GEMPAK subroutines. Part 2 contains GEMPLT subroutines to access the GEMPAK plotting package.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
meteorology software systems	Unclassified - Unlimited Subject Category 47

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	672	A99

NASA FORM 1626 OCT 86

For sale by the National Technical Information Service, Springfield, Virginia 22161-2171